UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA EN INFORMÁTICA

PROYECTO FIN DE CARRERA

BASES DE DATOS ESPACIO-TEMPORALES APLICADAS AL CONTROL DE VELOCIDADES

Tutor: Manuel Velasco de Diego

Autor: Ana María Rojas Barci

Noviembre, 2010

AGRADECIMIENTOS

A lo largo de este tiempo ha habido muchas personas a las que tengo que agradecer su apoyo y su constancia.

A mi familia, por estar ahí y por ser tan pacientes siempre conmigo. A mi padre que siempre me deja por imposible pero al cual le debo todo lo que soy. A mi madre, que con su constancia hace que no deje las cosas y que siempre tenga fuerzas para terminarlas. A mi hermana, que sin su apoyo y su comprensión nunca habría podido continuar. A mi cuñado, otro informático loco de los que el mundo está lleno.

A Nacho, que es la luz de mi vida. Sin ti sabes que no estaría donde estoy y no sería más que un reflejo de lo que nunca llegaría a ser. Te debo toda mi felicidad y mi futuro.

Por supuesto, a mis amigas Patricia y Nuria que sabían que un día llegaría este día y que me han animado y dado fuerzas todo este tiempo. Sólo decir que sin vosotras a mi lado mi vida no estaría completa.

A mis amigas de la universidad Irene, Rita, etc. que estoy segura que ya me dejaban por imposible pero que gracias a ellas siempre he podido sacar fuerzas de donde no las tenía. Aunque no nos veamos como antes me acuerdo siempre de vosotras.

Por último a mi tutor, Manuel Velasco, que sin su infinita paciencia nunca habría podido terminar el proyecto. Gracias por aguantarme todo este tiempo.

Se que me dejo muchas personas, de la universidad, del trabajo y demás, pero aunque no puedo dar las gracias personalmente a cada uno de ellas, han sido una parte muy importante y espero que lo sigan siendo.

Finalmente decir que una persona se forma gracias a las personas que la rodean, y yo he tenido la gran suerte de rodearme de personas, que aunque a lo mejor no cambian el mundo, sí que han cambiado el mío.

Muchas gracias por todo.

ÍNDICE

1 INTRODUCCIÓN

1.1 Descripción y objetivos del problema

El objetivo principal de este Proyecto de Fin de Carrera es llevar a la práctica las características de una base de datos espacio-temporal, y en particular, de una base de datos con objetos en movimiento dentro de un dominio concreto como es el de los vehículos de transporte dentro de una empresa. Las bases de datos espacio-temporales manejan cambios relacionados con el movimiento y las formas de los objetos a lo largo de su ciclo de vida.

En la resolución del problema se han utilizado geometrías propias de las bases de datos temporales, y tipos de datos propios de las bases de datos temporales. De esta forma se unifican las dos características de estas bases de datos.

Para intentar conseguir los objetivos propuestos en el dominio de nuestra aplicación, se ha optado por representar la bases de datos mediante un paquete de datos espacial, denominado Oracle Spatial, integrado en el SGBD de Oracle 10g añadiendo una extensión temporal al mismo.

Con el fin de facilitar el manejo de la base de datos, se ha implementado una interfaz que ayuda a comprender la estructura de la misma. Mediante esta interfaz se podrán realizar operaciones, tanto de inserción, borrado y consultas sobre la base de datos espacio-temporal.

2 ESTADO DEL ARTE

2.1 Bases de Datos Relacionales

Una base de datos relacional es un conjunto de dos o mas tablas estructuradas en registros (líneas) y campos (columnas), que se vinculan entre sí por un campo en común, en ambos casos posee las mismas características como por ejemplo el nombre de campo, tipo y longitud; a este campo generalmente se le denomina ID, identificador o clave. A esta manera de construir bases de datos se le denomina modelo relacional.

Estrictamente hablando el término se refiere a una colección específica de datos pero a menudo se le usa, en forma errónea como sinónimo del software usado para gestionar esa colección de datos. Ese software se conoce como sistema gestor de base de datos relacional o RDBMS (Relational Database Management System).

El modelo relacional es el más utilizado para modelar problemas reales y administrar datos dinámicamente. En 1970 fueron postuladas sus bases por Edgar Frank Codd, de los laboratorios IBM en San José (California).

Las reglas de Codd son las siguientes:

0. Todo sistema relacional debe manejar sus datos a través de sus capacidades relacionales exclusivamente.

1. **Representación de la información**: toda información en una base de datos relacional debe representarse explícitamente a nivel lógico, y de manera única, por medio de valores en tablas.

2. **Acceso garantizado**: todo dato debe ser accesible mediante una combinación de un nombre de tabla, un valor de su clave y el nombre de una columna.

3. **Tratamiento sistemático de valores nulos**: los valores nulos, información desconocida o inaplicable, han de ser tratados sistemáticamente por el sistema, el cual ha de ofrecer las facilidades necesarias para su tratamiento.

4. **Catálogo activo en línea basado en el modelo relacional**: la representación de la metainformación (descripción de la base de datos) debe ser igual a la de los otros datos, y su acceso debe poder realizarse por medio del mismo lenguaje relacional que se utiliza para los demás datos; es decir, el modelo de datos para la metainformación debe ser el relacional.

5. **Sublenguaje de datos completo**: debe existir un lenguaje que permita un complejo manejo de la base de datos:

   - ✓ definición de datos
   - ✓ definición de vistas
   - ✓ manipulación de datos
   - ✓ restricciones de integridad
   - ✓ autorizaciones
   - ✓ gestión de transacciones

6. **Actualización de vistas**: toda vista teóricamente actualizable debe poder ser actualizada por el sistema.

7. **Inserciones, modificaciones y eliminaciones de alto nivel**: todas las operaciones de manipulación de datos (consulta, inserción, modificación y borrado) deben operar sobre conjuntos de filas (lenguaje no navegacional).

8. **Independencia física de los datos**: el acceso lógico a los datos debe mantenerse incluso cuando cambien los métodos de acceso o la forma de almacenamiento.

9. **Independencia lógica de los datos**: los programas de aplicación no deben verse afectados por cambios en las tablas que estén permitidos teóricamente y preserven la información.

10. **Independencia de la integridad**: Las reglas de integridad de una base de datos deben ser definibles por medio del sublenguaje de datos relacional y han de almacenarse en el catálogo de la base de datos (metabase), no en los programas de aplicación.

11. **Independencia de la distribución**: Debe existir un sublenguaje de datos que pueda soportar bases de datos distribuidas sin alterar los programas de aplicación cuando se distribuyan los datos por primera vez o se redistribuyan éstos posteriormente.

12. **Regla de la no subversión**: Si un SGBD soporta un lenguaje de bajo nivel que permite el acceso fila a fila, éste no puede utilizarse para saltarse las reglas de integridad expresadas por medio del lenguaje de más alto nivel.

La idea fundamental de Codd fue el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas. Pese a esta teoría de las bases de datos relacionales, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es, pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario casual de la base de datos. La información puede ser recuperada o almacenada por medio de consultas que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más común para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Además, las bases de datos relacionales pasan por un proceso al que se le conoce como normalización de una base de datos, la cual es entendida como el proceso necesario para que una base de datos sea utilizada de manera óptima.

2.1.1 Diseño de una Base de Datos

En el diseño de una base de datos se debe realizar un modelo de datos que ayude a entender el significado de los datos y que facilite la comunicación en cuanto a los requisitos de información. La primera etapa es el diseño conceptual, en donde se construye un esquema de la información que maneja la empresa, independientemente de todas las consideraciones físicas. Después viene el diseño lógico, en el que el esquema anterior se transforma según el modelo de base de datos que se vaya a utilizar para implementar el sistema. Por último, en la etapa del diseño físico, se produce una descripción de la implementación de la base de datos en memoria secundaria.

El diseño de las aplicaciones, una fase que se debe llevar a cabo en paralelo con el diseño de la base de datos, está compuesta por dos actividades: el diseño de las transacciones y el diseño de las interfaces de usuario de informes y formularios.

2.1.1.1 El modelo conceptual

Un modelo conceptual es un conjunto de conceptos que permiten describir la realidad mediante representaciones lingüísticas y gráficas. Los modelos conceptuales deben poseer una serie de propiedades:

- Expresividad
- Simplicidad
- Minimalidad
- Formalidad

El modelo conceptual más utilizado es el modelo entidad-relación (E/R), que posee los siguientes conceptos:

- Entidades
- Relaciones
- Atributos
- Dominios de atributos

- Identificadores
- Jerarquías de generalización

2.1.1.2 El diseño lógico

Las dos fases de que consta el diseño lógico son la construcción y validación de los esquemas lógicos locales para cada vista de usuario, y la construcción y validación de un esquema lógico global. Cada una de estas fases consta de una serie de pasos.

Un paso importante es la conversión del esquema conceptual a un esquema lógico adecuado al modelo relacional. Para ello, se deben hacer algunas transformaciones: eliminar las relaciones de muchos a muchos, eliminar las relaciones complejas, eliminar las relaciones recursivas, eliminar las relaciones con atributos, eliminar los atributos multievaluados, reconsiderar las relaciones de uno a uno y eliminar las relaciones redundantes.

Los esquemas lógicos se pueden validar mediante la normalización y frente a las transacciones de los usuarios. La normalización se utiliza para mejorar el esquema, de modo que éste satisface ciertas restricciones que evitan la duplicidad de datos. La normalización garantiza que el esquema resultante está más próximo al modelo de la empresa, es consistente, tiene la mínima redundancia y la máxima estabilidad.

Las restricciones de integridad son las restricciones que se imponen para que la base de datos nunca llegue a un estado inconsistente. Hay cinco tipos de restricciones de integridad: datos requeridos, restricciones de dominio, integridad de entidades, integridad referencial y reglas de negocio.

Para garantizar la integridad referencial se debe especificar el comportamiento de las claves ajenas: si aceptan nulos y qué hacer cuando se borra la tupla a la que se hace referencia, o cuando se modifica el valor de su clave primaria.

2.1.1.3 El diseño físico

El diseño físico es el proceso de producir una descripción de la implementación de la base de datos en memoria secundaria. Describe las relaciones base y las estructuras de almacenamiento y métodos de acceso que se utilizarán para acceder a los datos de modo eficiente. El diseño de las relaciones base sólo se puede realizar cuando el diseñador conoce perfectamente toda la funcionalidad que presenta el SGBD que se vaya a utilizar.

El primer paso consiste en traducir el esquema lógico global de modo que pueda ser fácilmente implementado por el SGBD específico. A continuación, se escogen las organizaciones de ficheros más apropiadas para almacenar las relaciones base, y los métodos de acceso, basándose en el análisis de las transacciones que se van a ejecutar sobre la base de datos. Se puede considerar la introducción de redundancias controladas para mejorar las prestaciones. Otra tarea a realizar en este paso es estimar el espacio en disco.

La seguridad de la base de datos es fundamental, por lo que el siguiente paso consiste en diseñar las medidas de seguridad necesarias mediante la creación de vistas y el establecimiento de permisos para los usuarios.

El último paso del diseño físico consiste en monitorizar y afinar el sistema para obtener las mejores prestaciones y satisfacer los cambios que se puedan producir en los requisitos.

2.2 Bases de Datos Espaciales

2.2.1 Introducción

Una Base de Datos Espacial (Spatial Database) es un sistema administrador de bases de datos que maneja datos existentes en un espacio o datos espaciales. El espacio establece un marco de referencia para definir la localización y relación entre objetos. El que normalmente se utiliza es el espacio físico que es un dominio manipulable, perceptible y que sirve de referencia. La construcción de una base de datos geográfica implica un proceso de abstracción para pasar de la complejidad del mundo real a una representación simplificada que pueda ser procesada por el lenguaje de las computadoras actuales. Este proceso de abstracción tiene diversos niveles y

normalmente comienza con la concepción de la estructura de la base de datos, generalmente en capas; en esta fase, y dependiendo de la utilidad que se vaya a dar a la información a compilar, se seleccionan las capas temáticas a incluir.
La estructuración de la información espacial procedente del mundo real en capas conlleva cierto nivel de dificultad. En primer lugar, la necesidad de abstracción que requieren los computadores implica trabajar con primitivas básicas de dibujo, de tal forma que toda la complejidad de la realidad ha de ser reducida a puntos, líneas o polígonos. En segundo lugar, existen relaciones espaciales entre los objetos geográficos que el sistema no puede obviar; la topología, que en realidad es el método matemático-lógico usado para definir las relaciones espaciales entre los objetos geográficos puede llegar a ser muy compleja, ya que son muchos los elementos que interactúan sobre cada aspecto de la realidad.

Un modelo de datos geográfico es una abstracción del mundo real que emplea un conjunto de objetos dato, para soportar el despliegue de mapas, consultas, edición y análisis. Los datos geográficos, presentan la información en representaciones subjetivas a través de mapas y símbolos, que representan la geografía como formas geométricas, redes, superficies, ubicaciones e imágenes, a los cuales se les asignan sus respectivos atributos que los definen y describen.

2.2.2 Tipos de datos espaciales

Un dato espacial es una variable asociada a una localización del espacio. Normalmente se utilizan datos vectoriales, los cuales pueden ser expresados mediante tres tipos de objetos espaciales.
Los datos espaciales se dividen en:

- Puntos: Un punto representa un objeto del cual sólo interesa conocer su localización en el espacio. Su forma de representación más habitual es mediante dos coordenadas (x, y) en el espacio. Se encuentran determinados por las coordenadas terrestres medidas por latitud y longitud. Por ejemplo, ciudades, accidentes geográficos puntuales, hitos.
- Líneas: Una línea es comprendida como una secuencia de puntos y por tanto como una secuencia de coordenadas (x, y), normalmente representa conexiones

en el espacio. Debido a la forma esférica de la tierra también se las consideran como arcos. Líneas telefónicas, carreteras, ríos y vías de trenes son ejemplos de líneas geográficas

- Polígonos o Regiones: Figuras planas conectadas por distintas líneas u objetos cerrados que cubren un área determinada, como por ejemplo países, regiones o lagos. En la definición de las regiones, puede contemplarse la posibilidad de tener agujeros en su interior (R2), lo cual se conoce como región cóncava.

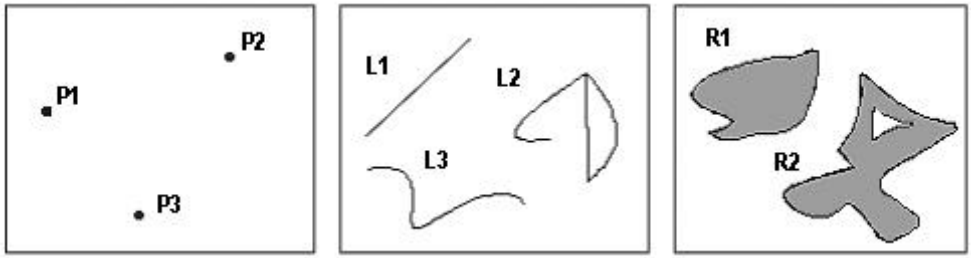

Figura 1: Entidades geométricas: Puntos, Líneas y Regiones

De esta forma la información sobre puntos, líneas y polígonos se almacena como una colección de coordenadas (x, y). La ubicación de una característica puntual, pueden describirse con un sólo punto (x, y). Las características lineales, pueden almacenarse como un conjunto de puntos de coordenadas (x, y). Las características poligonales, pueden almacenarse como un circuito cerrado de coordenadas. La otra forma de expresar datos espaciales es mediante rasterización, la cual, a través de una malla que permite asociar datos a una imagen; es decir, se pueden relacionar paquetes de información a los píxeles de una imagen digitalizada.

Los datos espaciales además se caracterizan por su naturaleza georreferenciada y multidireccional. La primera se refiere que la posición relativa o absoluta de cualquier elemento sobre el espacio contiene información valiosa, pues la localización debe considerarse explícitamente en cualquier análisis. Por multidireccional se entiende a que existen relaciones complejas no lineales, es decir que un elemento cualquiera se relaciona con su vecino y además con regiones lejanas, por lo que la relación entre todos los elementos no es unidireccional. Es decir, todos los elementos se relacionan entre sí, pero existe una relación más profunda entre los elementos más cercanos.

Los datos espaciales poseen algunas características que hacen inapropiada la utilización de los Sistemas Manejadores de Bases de Datos (DBMS) tradicionales, motivando la

aparición de las bases de datos espaciales con objeto de permitir su gestión de una forma eficiente. Algunas de esas características son:

- La complejidad de las estructuras de datos necesarias para su almacenamiento, como es el caso de un polígono, hace inviable la posibilidad de emplear los tipos de datos tradicionales, dado que se precisa registrar un conjunto de puntos que lo delimiten, puntos que pueden ser d-dimensionales. Esto hace las DBMS tradicionales, como pueden ser las relacionales con un tamaño de tupla fijo, inapropiadas en este dominio. Además la necesidad de contemplar toda una serie de operaciones, como la intersección o la unión que posterior mente (ver apartado 3.2), que necesitan una implementación eficiente en el DBMS, hace inevitable la extensión o definición de un modelo que les de soporte.
- La necesidad de ofrecer estructuras de datos que permitan tanto la inserción, como el borrado y actualización de una forma robusta y con un rendimiento elevado, sin que el coste de procesamiento sea demasiado elevado.
- El elevado volumen de datos con el que se suele trabajar en aplicaciones que tratan con sistemas VLSI o con información geográfica como los Sistemas de Información Geográfica, pone de manifiesto la necesidad de emplear sistemas de almacenamiento secundario que permitan su almacenamiento y recuperación de forma eficiente.
- La carencia de un álgebra espacial estándar, implica la imposibilidad de utilización de un conjunto de operaciones cuya semántica esté claramente definida. En su lugar, se encuentran diversas propuestas en este campo que en muchas ocasiones dependen demasiado del dominio de aplicación.
- La dimensionalidad de los datos espaciales supone a su vez otro problema, debido a la inexistencia de un orden total que permita aplicar los predicados existentes en las bases de datos tradicionales. Este orden se refiere a la imposibilidad de, por ejemplo, realizar una consulta según los objetos espaciales sean mayores que uno dado. No existe una definición del concepto mayor sobre los datos espaciales.

– El problema de cierre de las operaciones espaciales aparece con demasiada frecuencia, debido a los diferentes tipos sobre los que se pueden aplicar. Tomando como ejemplo el caso de la intersección entre dos objetos espaciales, el resultado podría ser tanto un polígono, como una línea o un punto, o una serie de estos objetos. En muchas aproximaciones se desprecia parte del conjunto de posibles soluciones debido a deficiencias en el modelo en el que se basa.

Independientemente del modelo de DBMS que se emplee sea orientado a objetos, relacional u objeto-relacional, se trata de realizar una extensión de en su arquitectura de forma que soporte la representación de estos objetos geométricos. Esta extensión significa, en primer lugar, la modificación del modelo de datos para permitir la inclusión de nuevos *tipos abstractos de datos espaciales*, tales como puntos, líneas o regiones, que permitan recoger la geometría de las entidades que queremos almacenar y, en segundo lugar, el conjunto de operaciones que definen su comportamiento. Junto a la inclusión de estos tipos de datos es imprescindible la extensión de los lenguajes tanto de consulta como de definición, que permitan manipular datos espaciales, así como un conjunto de técnicas de indexación adaptadas al tipo de datos con el que se esté trabajando, que permitan obtener un alto rendimiento del sistema.

2.2.3 Modelos de datos espaciales

El propósito de cualquier modelo de datos, es proporcionar un medio formal con el que representar un determinado tipo de información, así como un conjunto de operaciones que permitan su manipulación. Esa representación trata de abstraer un determinado aspecto de la realidad. En este caso, el espacio d-dimensional que es por naturaleza infinito y más concretamente, el espacio Euclídeo en la mayoría de las aproximaciones. Desafortunadamente, los modelos de datos que se presentarán seguidamente, dan una representación finita de esos infinitos e incluso innumerables conjuntos de puntos que componen los objetos espaciales registrados en las bases de datos, llevando en ocasiones a una pérdida inevitable de información.

Un modelo de datos es considerado generalmente como una abstracción, que incorpora solamente aquellas propiedades pensadas como relevantes a la aplicación o aplicaciones con las que se está tratando, usualmente una conceptualización humana de la realidad

sin connotaciones tecnológicas en su mayor nivel de abstracción, es decir, sin considerar donde será implementado. Pero, la implementación del mismo conlleva generalmente a tomar decisiones en las que se prima el rendimiento frente a la precisión en la representación. Esta decisión impacta directamente en las características de almacenamiento, manipulación y recuperación de las estructuras tanto física como de los datos en sí mismos. Se hace necesario examinar esos temas utilizando un conjunto de criterios basados en el uso, de forma que la calidad total o la idoneidad de un modelo de datos específico puedan ser evaluados dentro de un contexto particular.

Los criterios generales son:

- *completitud*, que puede plantearse en términos de la proporción de todas las entidades y relaciones de un fenómeno particular, existentes en la realidad y que son representadas en el modelo.
- *robustez*, es el grado con que el modelo de datos puede acomodarse a circunstancias especiales o poco usuales, tales como un polígono como un agujero en su interior.
- *versatilidad*, permitiendo contemplar situaciones no recogidas inicialmente en el modelo.
- *eficiencia*, incluye tanto la compactación, es decir, eficiencia de almacenamiento, como la velocidad de uso entendida como eficiencia temporal.
- *facilidad de generación*, es la cantidad de esfuerzo necesario para convertir datos de un formato a algún otro requerido por el modelo de datos.

El grado de variación de cada uno de estos factores ha de tomarse en consideración para cualquier campo de aplicación. La importancia relativa de cada factor es una función de los tipos de datos particulares a ser empleados y de los requisitos operacionales totales del sistema. Por ejemplo, si la base de datos a ser generada tiene un volumen elevado y tiene que ofrecer un rendimiento en un contexto interactivo, sería adecuado un compromiso entre los tres primeros factores, dado que la eficiencia total y la facilidad de generación predominarían.

Es posible medir cuantitativamente el rendimiento de varios de estos factores como la velocidad y la eficiencia en espacio para un modelo de datos particular. No sin embargo, proporcionar medidas cuantitativas para los factores más abstractos como completitud,

robustez o versatilidad. Este hecho combinado con el escaso conocimiento sobre las características de rendimiento de un amplio rango algoritmos de procesamiento espacial así como su interacción con otros algoritmos y modelos de datos, indica que el proceso de modelado de datos es mucho mas un arte que una ciencia. Para ellos, la experiencia y la intuición se mantendrán como factores clave en la interpretación de las especificaciones de requisitos de sistemas poco definidos y la construcción de modelos de datos satisfactorios, particularmente para sistemas de información geográfica integrados y completos.

En la definición de estos modelos de datos encontramos dos alternativas claramente diferenciadas. La primera de ellas realiza una definición abstracta de los tipos de datos de forma que permita su posterior integración en el DBMS con independencia de si éste es relacional, objetual o de cualquier otro tipo. En cambio, la segunda alternativa hace uso del modelo utilizado por el sistema gestor que pretende extender para realizar su definición. Existe un punto en común a todas las alternativas que es la semántica asociada a los tipos de datos que se pretenden incorporar.

2.2.3.1 Modelo de datos Ráster

En este tipo de modelo la información geométrica es intencionalmente descrita por un número finito de puntos ráster. La semántica en este modelo es que el número infinito de puntos en el entorno de un punto ráster p tienen sus mismas propiedades. Los puntos ráster están uniformemente distribuidos sobre el objeto que está siendo representado. Aunque el modelo es bastante intuitivo, presenta algunos problemas relacionados con la distribución, en ocasiones heterogénea, de las propiedades relevantes, por ejemplo una recta que no se ajusta a los puntos que conforman el ráster, dando lugar a problemas con el cierre de las operaciones.

Una de las características de este modelo es la extensibilidad de la arquitectura del sistema, dada la posibilidad de poder definir nuevos tipos de acuerdo a las necesidades del dominio concreto donde se vayan a aplicar. Incluye también diferentes operaciones que permiten la conversión entre diferentes tipos, así como completitud en cuanto al grado de objetos espaciales que son representables mediante este modelo. Así mismo, es eficiente tanto en términos de utilización de almacenamiento, debido a la escasa duplicación de información, como temporal, gracias al cálculo de las intersecciones en el momento de inserción de los objetos espaciales.

El modelo de datos Ráster se utiliza en los siguientes casos:

- Rápida y correcta superposición de mapas y, en general, para el análisis espacial.
- Modelado y análisis de superficies.

2.2.3.2 Modelo de datos Polinomial

El modelo de datos con restricciones introducido por Kanellalis en 1995 trata de mostrar un paradigma para la representación de todos los tipos de datos desde un marco unificado. Las restricciones lineales sobre los números racionales han demostrado, mediante la representación modo vector, su idoneidad para la representación de datos espaciales. Este formato de representación se enfrenta con una representación en la que se emplean los bordes de un objeto para representar los infinitos puntos que lo componen, y que conlleva la inexistencia de un estándar que permita aportar una solución única a este problema.

La idea básica de este modelo es representar los objetos espaciales como colecciones infinitas de puntos en el espacio de los racionales (Q2) que satisfacen una formula de primer orden. Por ejemplo, un polígono convexo, que es la intersección de un conjunto de medios planos, es definido mediante la conjunción de las inigualdades que definen cada uno de ellos. Un polígono no convexo es definido mediante la unión de un conjunto de polígonos convexos.

Aunque no restringe el tipo de objetos que pueden ser almacenados en la base de datos, sí en cambio obliga a una descomposición del espacio en componentes convexos en el momento de la carga de la información. Aunque esto conlleva un mayor coste de procesamiento inicial, simplifica enormemente cualquier consulta posterior. Este tipo de representación se conoce como DNF (Disjunctive Normal Form, Figura 2). En cambio, ese procesamiento puede ser evitado si se permite la definición de polígonos con agujeros, es decir, la representación conocida como CHNF (Convex with Holes Normal Form, Figura 2).

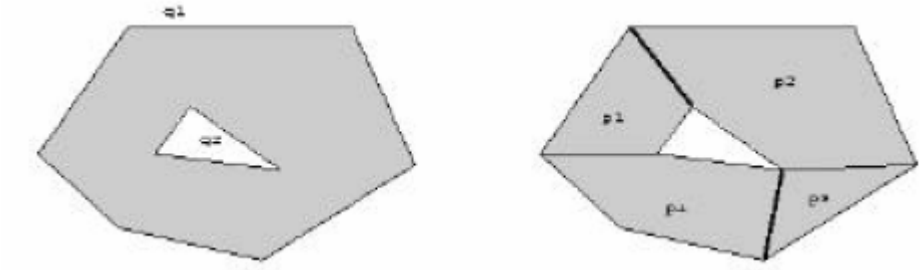

Figura 2: Representación DNF y CHNF de un polígono con agujeros

2.2.3.3 Modelo de datos Topológico

El interés principal de este tipo de modelo es ofrecer una base forma para aquellas consultas en la base de datos con referencia a las relaciones topológicas de los objetos, tales como adyacencia o solapamiento. Una característica inherente de las bases de datos que soportan este tipo de relaciones es la equivalencia entre las bases de datos que son generadas mediante deformaciones topológicas, es decir, son equivalentes topológicamente.

Este tipo de modelo se aplica generalmente en dominios enfocados a modelar relaciones entre los objetos. Un caso de un dominio de aplicación es un sistema de control de tráfico donde, una de sus necesidades básicas es registrar los enlaces entre diferentes carreteras. Estas relaciones permiten, por ejemplo, determinar las posibles rutas alternativas.

2.3 Bases de Datos Temporales

2.3.1 Introducción

En 1998 R. T. Snodgrass ofreció una clara definición de lo que una Base de Datos Temporal es. Según Snodgrass una Base de Datos Temporal es una base de datos que soporta algunos aspectos de tiempo, no contando el tiempo definido por el usuario.

El “Tiempo definido por el usuario” es algún aspecto de tiempo, el cual no es reconocido por el sistema administrador de la base de datos siendo un tipo de dato especial. El "Tiempo definido por el usuario" se define como un atributo sin interpretación de dominio fecha y tiempo paralelo a aquellos dominios como dinero y enteros.

Tradicionalmente, las bases de datos tratan de representar una visión de la información referente al mundo real. Dicha información tiene una validez determinada en un instante determinado. Por lo que si esta realidad se ve alterada en algún momento el sistema gestor (DBMS) ha de reflejar este cambio mediante actualizaciones o borrados en la base de datos, según sea la naturaleza del cambio.

Esta manera de tratar las modificaciones conlleva que el estado previo de la información se pierda o se desprecie. Este tipo de bases de datos se conoce como base de datos "snapshot", debido a que ofrecen una fotografía de la realidad en un momento concreto**.**
La motivación de gestionar así los cambios en la información proviene de la capacidad, siempre limitada, de almacenamiento o de carencias en el modelo de datos que permitan capturar esa evolución de la información. La aparición de mejoras en la tecnología ha permitido utilizar mayores y más rápidos sistemas para almacenamiento de la información, erradicando de las limitaciones del sistema entre las restricciones que soportan tal decisión.
A estas mejoras en la tecnología se une la aparición de las bases de datos temporales, que se caracterizan, esencialmente, por incorporar la representación de aspectos temporales en el modelo de datos, con una semántica especial junto con las facilidades en el sistema gestor necesarias para su manipulación.

2.3.2 Conceptos generales

Una base de datos temporal es aquella que contiene datos históricos en lugar de, o así como, datos actuales, es decir, situaciones en las que es necesario reflejar la validez temporal de la información, durante qué intervalo o instante de tiempo se consideró o fue considerada como válida (C. J. Date, An Introduction to Database Systems. October 1999).
Así, un ejemplo podría darse al intentar reflejar la información temporal referente a la propiedad de una parcela L. Podemos encontrar diferentes alternativas para registrar esa evolución:
1. La parcela L fue adquirida por O el 1 de Enero de 1999.
2. La parcela L ha pertenecido a O desde el 1 de Enero de 1999 hasta la fecha.
Cada una de las sentencias anteriores proporciona una interpretación diferente sobre el estado o la situación de propiedad en que se encuentra la parcela L. En la primera, nos informa acerca del momento concreto en que la parcela L pasó a ser propiedad de O; no proporciona información sobre si la parcela pertenece o no en la actualidad a O, solo indica un instante temporal. En la segunda, además, se representa el período o intervalo de tiempo durante el cual la parcela L ha pertenecido a O. En ambas, los datos tienen una referencia temporal, introducida mediante la fecha "1 de Enero de 1999", que

recibe, en la literatura inglesa, el nombre de timestamp, que marca en ambos casos un instante temporal. El timestamp nos permite incluir una marca temporal sobre un momento de validez de la información. Podemos apreciar igualmente, que la información referente a la parcela, como por ejemplo su propietario, extensión, catalogación, puede estar asociada a cualquier número de eventos o de intervalos a lo largo del tiempo.

Basándonos en el ejemplo anterior, podemos introducir dos conceptos claves en el campo de las bases de datos temporales: tiempo de validez (valid time) y tiempo de transacción (transaction time).

El tiempo de validez expresa el tiempo durante el cual una cierta proposición es cierta, es decir, según el caso anterior deberíamos almacenar que la validez de la información relativa a la propiedad de la parcela es: "desde el 1 de Enero de 1999 hasta hoy".

El tiempo de transacción indica el tiempo en que una proposición aparece reflejada en la base de datos como cierta, el momento en que incorporamos esa información en la base de datos.

El primero de ellos puede ser actualizado por el usuario para reflejar un cambio en la proposición, en cambio, el segundo puede ser únicamente modificado por el sistema gestor, cualquier cambio que un usuario realice sobre la información quedaría reflejado mediante un nuevo tiempo de transacción que marcaría el momento en que hizo la modificación.

El tiempo de validez y de transacción no debe confundirse con el tiempo de usuario (user time). Éste representa el empleo, por parte del usuario de un tipo de datos estándar (date) de un modo similar al empleo de los enteros o los reales, para representar alguna información temporal. Por ejemplo, en una base de datos de personal, al introducir una información como la fecha de alta de un usuario o su fecha de nacimiento se está reflejando una información temporal, no su período o instante de validez. En este caso, no se ofrece un soporte especial del lenguaje de consulta como en el caso del tiempo de validez o de transacción.

La introducción del tiempo de validez y de transacción, ha permitido la definición de los operadores: valid timeslice y transaction timeslice. Ambos toman como argumento una relación, con una referencia temporal ya sea de validez o de transacción, dependiendo de si se trata del primer o del segundo operador, y un valor temporal que indica un momento o instante sobre el que queremos realizar la operación. De esta forma el

operador valid timeslice devolverá el estado de la relación válida en ese instante, es decir, toda aquella información que contuviera esa relación valida en ese momento. El uso del operador transaction timeslice, permitirá obtener aquella información que era actual en ese momento.

De igual forma que el tiempo de transacción y de validez son aplicables sobre una información almacenada en la base de datos, también es posible aplicar estos tiempos a una relación. Es decir, permitir que pueda reflejarse cuando una relación ha sido incluida en el esquema o cuando se han efectuado modificaciones sobre ella en el primer caso, o delimitar, en el caso de tiempo de validez, en que instante o durante que intervalo de tiempo la relación ha sido válida. Marcando por tanto una diferencia sobre las relaciones snapshot, que podemos encontrar en los sistemas de gestión tradicionales, en las que una relación no viene cualificada por ningún atributo temporal.

Las DBMS comerciales almacenan sólo un estado del mundo real, usualmente el estado más reciente. Aquellas bases de datos usualmente son llamadas "snapshot databases" cuyo contexto de tiempo valido y tiempo de transacción es mostrado en la siguiente figura:

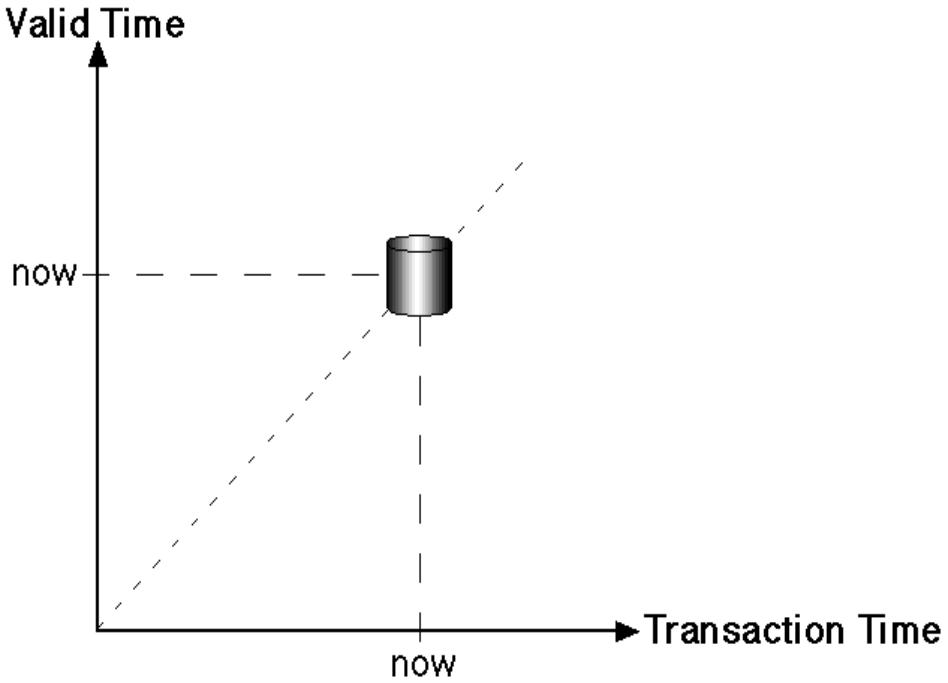

Figura 3: Snapshot Databases

R. T. Snodgrass clasifica las bases de datos en estáticas, históricas, rollback o bitemporales, según como se contemplen, o no, los conceptos anteriores dentro del esquema.

Las bases de datos estáticas no contemplan ni el tiempo de validez ni de transacción, no almacenan ninguna de estas referencias temporales en el modelo que emplean.

Las bases de datos históricas contemplan únicamente el tiempo de validez, almacenan la información que conocemos como válida, tanto presente como pasada o futura, pero los cambios producidos en la información (como ha sido la evolución de las actualizaciones que se han realizado) no quedan almacenados.

Las bases de datos rollback, en cambio, contienen exclusivamente tiempo de transacción, nos permiten devolver la base de datos a un estado anterior en caso de que sea necesario, pero no permiten conocer durante que intervalo de tiempo ha permanecido como valida una determinada información.

Por último, las bitemporales soportan tanto el tiempo de transacción como el de validez, por lo que nos dan una visión más precisa de la evolución que ha sufrido la información tanto sobre sus intervalos de validez como de sus actualizaciones, al contemplar los diferentes estados por los que pasó la información, y permitir su recuperación. Los estados almacenados en una Base de Datos bitemporal están representados en la siguiente figura:

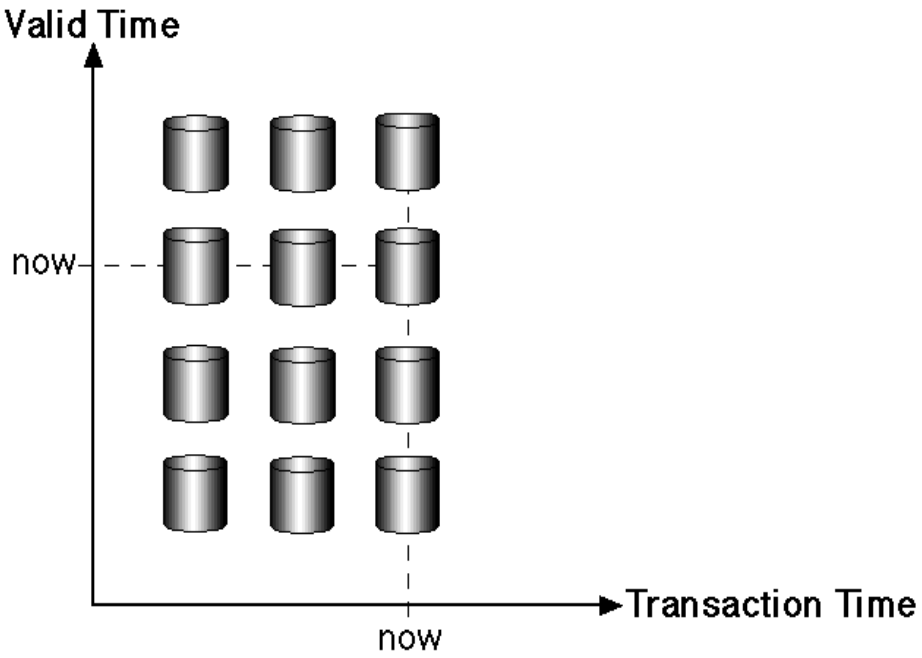

Figura 4: Base de Datos Bitemporal

Otro concepto, que afecta especialmente a la semántica de las operaciones y que ha de ser incluido es la definición de Crono o quantum temporal. Éste hace referencia a la duración de tiempo más corta, al instante temporal más breve, soportado por el DBMS. Un quantum determina un subintervalo de tiempo, con una duración fija a lo largo de la línea temporal.

2.3.3 Tipos de datos más comunes de una Base de Datos Temporal

Una base de datos temporal soporta tres tipos de datos más comunes: dato temporal, dato estático y dato instantáneo. Un elemento de importancia a considerar es que los distintos tipos de datos deben ser transparentes para el usuario, pues el no necesita saber que esta trabajado con un temporal, un estático o con uno instantáneo. Una meta de las bases de datos temporales debe ser proveer el mismo tratamiento para datos temporales, estáticos e instantáneos. Para este fin, las bases de datos temporales deberían tratar todos los datos como una forma de dato temporal (Gadia y Nair, 1993).

El tipo de dato temporal es el más importante de la base de datos temporal. Shashi Adia y Sunil S. Nair identifican el tipo de dato temporal como "una unión finita de intervalos". Ellos llaman este tipo de dato un "elemento temporal". El tipo de dato temporal o elemento provee la base, para que una base de datos temporal sea construida. Se puede distinguir entre dos tipos de datos temporales:

- ✓ Instante: en un instante de tiempo, ocurre algún hecho
- ✓ Intervalo: durante un intervalo de tiempo, algo que ha ocurrido se mantiene. Su definición consiste de un punto de inicio y otro de fin, así como un tipo sobre el que se construye el intervalo, es decir, si se trata de un intervalo de enteros, fechas, etc.

El empleo de intervalos supone, entre otras ventajas, una formulación más sencilla de las consultas que en el caso de tener que tratar, por separado, ambos extremos del intervalo o imponer restricciones.

El tipo de dato estático es definido como "una constante definida sobre todo el universo de tiempo" [Gadia y Nair 1993] En otras palabras, la validación de un valor estático definido como "por siempre". En contraste un valor de tipo dato temporal es valido para

un periodo o intervalo específico de tiempo, y un valor de tipo de dato instantáneo es sólo valido para el instante actual (NOW) Otra forma de expresar la validación de un dato estático, es decir, que su dominio es eterno. Sin un periodo de validación es finito, el tipo de dato estático no necesita una marca de tiempo.

En una base de datos clásica, todo dato es del tipo instantáneo (Ozsoyoglu y Snodgrass 1995, Gadia y Nair 1993, Navathe y Ahmed 1993). Todos los datos en una base clásica son validos para el instante de tiempo en el cual éste existe. Cuando las tuplas son actualizadas, los nuevos valores reemplazan los viejos, los cuales son borrados y un nuevo valor instantáneo es creado. Desgraciadamente, con el dato de tipo instantáneo, la historia de los cambios de los datos se pierde.

Por otro lado, algunos datos no necesitan incluir información de tiempo variante, y este es el tipo de dato que debería ser del tipo instantáneo. Debido a esto, la base de datos temporal también debiera soportar el tipo de dato instantáneo. Otra razón para soportar el tipo de dato instantáneo, es proveer una suave transición desde la base clásica a la temporal.

2.3.4 Granularidad

Para representar los datos a diferentes niveles temporales de detalle, se considera la definición de granularidad temporal, comúnmente adoptada por la comunidad. La granularidad temporal expresa una restricción en la cual la máxima frecuencia de actualización de un atributo es la granularidad elegida. Se asume que el dominio del tiempo es un conjunto discreto de instantes de tiempo, en los que una relación de orden total es definida. Una granularidad es una partición, posiblemente no total, del dominio del tiempo.

La granularidad temporal es una medida de tiempo que puede ser expresada en días, minutos, segundos, etc. Su importancia deriva en como influye en la semántica de las operaciones temporales. A su vez, dado un conjunto de granularidades existen unas relaciones de equivalencia entre ellas, por ejemplo, la granularidad semanas es más fina que la granularidad meses y menos fina que la granularidad días. Estas relaciones de equivalencia entre diferentes granularidades deben ser especificadas en la base de datos. Por otro lado, la granularidad de la información se puede ajustar, moviéndose de algo más fino (fino) a datos más grueso (coarser), a esto se le conoce como rollup. Si ocurre lo opuesto, ir de algo grueso a algo más fino ocurre un drill down.

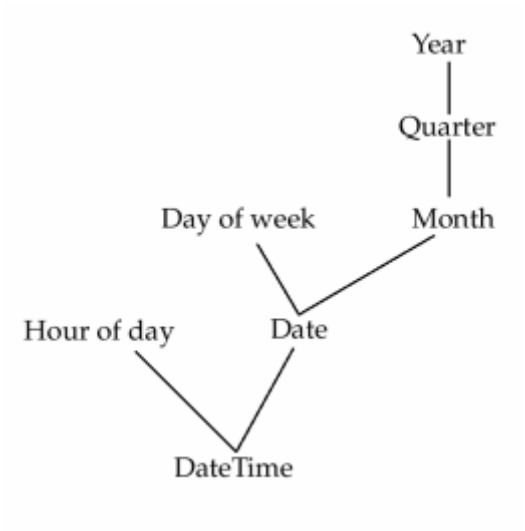

Figura 5: Equivalencia de granularidades

2.3.5 Extensión al Modelo Relacional

Como sabemos, el modelo relacional, y por ende, las bases de datos relacionales, presentan una imagen actual de los datos que almacenan. Esto se aprecia a través de los procesos posibles a efectuar sobre los datos. Por ejemplo, una actualización sobrescribe los datos existentes en una base de datos relacional, lo cual es contrario a la concepción de bases de datos temporales, que por su parte agregarían la nueva información sin eliminar la existente.

Podemos distinguir dos tipos de extensiones del modelo relacional, en función de la forma de incluir el tiempo, conocido como marca temporal (timestamp), como información relevante:

- ✓ A nivel de tupla
- ✓ A nivel de atributo

Con la primera opción (nivel de tupla), se indica que la validez de una tupla viene determinada por un atributo temporal, que refleja el instante de comienzo y fin de ese período de validez, pudiendo utilizarse un atributo perteneciente al tipo intervalo definido anteriormente. Esta forma de introducir la información temporal es la más común en el modelo relacional, ello puede deberse a que obedece la restricción impuesta por la primera forma normal (1FN), que consiste en la no inclusión de atributos multivaluados. Típicamente se utilizan los atributos "desde" y "hasta" para representar intervalos de tiempo, que según el modelo, podrán soportar la inclusión del futuro como dato válido, ya sea explícito o variable (∞, forever, uc: until changed); además, también puede limitarse de un intervalo, a un punto del tiempo.

Una consecuencia de esta opción, es que una entidad puede estar representada por varias tuplas en la base de datos, lo que, además de impedir una representación uno a uno de la realidad, hace posible la existencia de información redundante, dado que algunos atributos varían con mayor frecuencia que otros.

Otra desventaja, que es propia de que el modelo relacional temporal herede la 1FN, es la de no permitir una evolución de la identidad de las entidades. Por ejemplo, si la clave primaria de una entidad concreta, cambiara en el tiempo, la inclusión de este cambio, correspondería a una nueva tupla, lo que no refleja el cambio de identidad, sino que se considera como una nueva tupla independiente de la anterior.

La segunda alternativa (a nivel de atributo), se incluye la información temporal como parte del atributo, es decir, tanto el valor de la marca de tiempo (timestamp), como la entrada a la que éste hace referencia se almacenan en un mismo atributo de forma anidada. Al considerar en un mismo atributo la información temporal (tiempo de transacción y/o de validez), se disminuye la redundancia de información. Sin embargo, esto no evita los posibles problemas de actualización que pueden aparecer en estos casos, debido a que los valores anidados son afectados por la propagación de las modificaciones. Por lo tanto, se trata de un modelo que no cumple la primera forma normal, cuyas consecuencias lo hacen difícil de implementar.

Sin embargo, este enfoque presenta ciertas ventajas. Por medio de atributos multivaluados, un objeto de la realidad puede estar representado por sólo una tupla en la base de datos, con lo que se consigue una correspondencia uno a uno entre la base de datos y la realidad modelada. Esta propiedad aumenta el poder de modelado.

Por otra parte, y contrariamente a la primera opción, este modelo permite reflejar la evolución de atributos en el tiempo, con la condición de que todos ellos, estén bien definidos durante el tiempo en que son considerados.

Por último, y con respecto a la evolución de las entidades, tenemos que el modelo simplifica el mapeo de los cambios producidos, ya que se verán alterados sólo los atributos involucrados, pues en ellos se inserta el nuevo valor y su marca temporal. De esta manera, se evita la posible redundancia vista en la primera opción.

2.4 Bases de Datos Espacio-Temporales

El espacio y el tiempo son dos atributos propios de cualquier objeto del mundo real. De esta forma un objeto se caracteriza por su posición y el área o volumen que ocupa (en un espacio de referencia) en cualquier punto del tiempo. Estos objetos conforman el tipo de datos espacio-temporales los cuales requieren ser manejados por aplicaciones informáticas.

Inicialmente, las investigaciones sobre bases de datos trataban por separado los modelos de datos espaciales de los temporales. Fue en los años 90 cuando las Bases de Datos Espacio-Temporales se volvieron un área de investigación activa. A partir de entonces las Bases de Datos Espacio-Temporales tratan con aplicaciones en las que se integra conjuntamente el tiempo y el espacio.

La representación y gestión de las Bases de Datos Espacio-Temporales pretende dar soporte a aplicaciones como el control de tráfico de una ciudad, el tráfico aéreo en un aeropuerto o el análisis de fenómenos meteorológicos, donde es importante saber la variación en el tiempo de un objeto representado a través de una geometría. Hasta ahora, los Sistemas de Información Geográfica (GIS) al tratar la información espacial y no espacial de una manera heterogénea (sistemas de ficheros y bases de datos, respectivamente), han resultado poco eficientes para el mantenimiento de la integridad entre los datos. La utilización de las bases de datos es una aproximación que disminuye este problema.

Los componentes básicos de las bases de datos espacio-temporales son objetos espaciales, que son conjuntos finitos de puntos en un espacio (Pfoser & Tryfona, 2001). Desde una perspectiva temporal, las propiedades y relaciones son consideradas hechos de objetos y, por lo tanto, pueden tener asignados valores de verdad.

Una base de datos espacial registra información de puntos, líneas y regiones, sobre su localización y su geometría, e incluso en algunos casos, como en el modelo topológico podía reflejar las relaciones topológicas sobre los diferentes objetos, permitiendo realizar consultas sobre caminos más cortos, alternativas, etc. Sin embargo, ninguna de las anteriores aproximaciones, en el área de las bases de datos espaciales, se contempla la posibilidad de que esos objetos cambien su posición a lo largo del tiempo. Tampoco se presentan soluciones ante situaciones en las que la geometría de los objetos cambie ante determinados eventos.

El tiempo de existencia está asociado únicamente con la existencia de un objeto, y no con su posición. Para el tiempo, diferentes tipos de modelos están dados por puntos de tiempo e intervalos de tiempo. Un punto de tiempo es un instante en el tiempo, mientras que un intervalo de tiempo está definido por un punto de tiempo inicial y uno final. Una base de datos espacio-temporal puede almacenar eventos o estados. Un evento ocurre usualmente en un punto de tiempo específico, sin duración.

Existe por tanto la necesidad real en la que es preciso contemplar la gestión del espacio y del tiempo con un mismo enfoque integrando las características de ambos como soporte para la gestión de este tipo de información. Es necesario un conjunto de herramientas que permitan un uso y gestión eficiente de la llamada información espacio-temporal.

La introducción de información espacio-temporal, no se limita únicamente a incluir unos atributos temporales que permitan registrar ese cambio, sino que necesita la definición de nuevos modelos de datos que permitan recoger la semántica de ese cambio, junto con lenguajes y técnicas de indexación que permitan la recuperación y manipulación de una forma eficiente, de este tipo de información. Estas necesidades ponen de manifiesto las carencias no solo de los DBMS tradicionales, sino también de las bases de datos espaciales para tratar con este tipo de información.

2.4.1 Modelos de datos espacio-temporales

Los modelos espacio-temporales se caracterizan por tratar objetos cuya geometría cambia a lo largo del tiempo, es decir, tienen capacidad de gestionar geometrías en cambio continuo. Existen diferentes conceptos que deben tenerse en cuenta a la hora de establecer una posible clasificación entre los diferentes modelos.

En primer lugar, en el dominio espacial, la elección de la representación para el almacenamiento de datos espaciales. En la actualidad existen dos aproximaciones. La primera de ellas representación ráster en la que el espacio es fraccionado en una cuadrícula de forma que cada celda puede ser direccionada por su posición. La ventaja que ofrece este tipo de representación es la sencillez de cualquiera de las operaciones geométricas, al tratar de determinar la intersección de dos objetos en el espacio, su unión, etc. Sin embargo, su utilización plantea problemas de eficiencia tanto temporal como de almacenamiento, puesto que se necesita una gran cantidad de tiempo para procesar grandes volúmenes de información, sobre todo al trabajar con objetos de cierta resolución. La segunda, emplea la representación vectorial, según la cual el objeto espacial se representa a partir de una serie de puntos. Esta representación permite paliar los problemas antes comentados tanto respecto a la eficiencia temporal como de almacenamiento. Sin embargo, la complejidad de implementación de cualquiera de las operaciones que se defina es mucho mayor.

Otro aspecto a considerar se centra en el dominio temporal. Se trata en este caso de considerar como se introduce esa información relativa al tiempo dentro del modelo. Como ya se ha explicado, existen dos conceptos básicos que son tiempo de validez y tiempo de transacción. La forma en que estos conceptos son recogidos, o no, dentro del modelo permitirá dar diferentes semánticas a los tipos definidos, y que, a su vez, permitirá establecer diferencias entre los distintos modelos que se comentaran seguidamente.

Otro aspecto a tener en cuenta es si el modelo bajo estudio permite capturar las nociones de movimiento y de cambio. Con el movimiento, se trata de reflejar alteraciones en la posición en el espacio a lo largo del tiempo. Por el contrario, el concepto de cambio se ocupa de recoger como el objeto como sufre una transformación en su extensión. Ambas nociones se ven afectadas por cómo se produce esa evolución, es decir, si ésta se contempla de forma discreta, con lo que podría emplearse el espacio de los enteros para

realizar su representación, o bien, es continua con lo que deberían emplearse los reales o racionales para recogerla.

Relacionado con los conceptos de cambio y movimiento, otro aspecto que influye en la clasificación se refiere a la capacidad del modelo para realizar consultas sobre esa evolución, es decir, la posibilidad de contemplar la semántica que conlleva la alteración de la posición y de la extensión.

2.4.1.1 Modelo de datos Snapshot

G. Langran, en "A Framework for Temporal Geographic Information Systems" en 1998 presenta una de las aproximaciones más simples en las que la dimensión temporal está basada en un modelo discreto y lineal del tiempo, empleando una estructura ráster para registrar la información espacial. El dominio temporal se incorpora en el modelo mediante el tiempo de validez únicamente, pero con diferentes con diferentes granularidades.

La información espacial se incorpora mediante un conjunto de capas, en las que cada una de ellas representa una colección de información cuya validez es homogénea, es decir, toda la información que se encuentra recogida en una capa es valida durante el mismo intervalo temporal. La información temporal se incorpora como una marca temporal sobre cada una de esas capas, es decir, se considera un atributo de la información espacial.

Se presenta como el modelo más sencillo para representar la información espacio-temporal, pero con serias limitaciones en cuanto a las consultas relacionadas con la evolución temporal. Es importante recalcar que no existe ninguna relación entre las diferentes capas; en su lugar, cada capa recoge la información valida en un momento determinado. Por lo que, para detectar si las dos capas han evolucionado exige una comparación exhaustiva de las mismas, impidiendo que el modelo refleje claramente esa evolución.

Este modelo, a su vez, presenta ciertas desventajas:

- ✓ Rendimiento en el almacenamiento. Para cada uno de los cambios que sufre la información se ha de recoger toda la información, por la propia definición del modelo, tanto aquella que ha sufrido una evolución como la que no, con la consiguiente duplicación de la información.

- ✓ Sólo ofrece la posibilidad de registrar el tiempo de validez, no el tiempo de transacción. Debido a esto, sólo es posible recuperar la información a un estado previo teniendo en cuenta los instantes de modificación del usuario.
- ✓ No registra el movimiento de los objetos. Debido a la gran cantidad de snapshot que tendría que almacenar.

2.4.1.2 Modelo de datos orientado a eventos

Un modelo de datos orientado se caracteriza por representar los cambios más que las entidades, de manera que los diferentes estados por los que ha pasado una entidad pueden ser recuperados rastreando los cambios hasta alcanzar el instante deseado. Concretamente se aproxima a las bases de datos rollback dado que se utiliza el tiempo de transacción para marcar la entrada de la entidad espacial en la base de datos, de forma que cuando ésta sufre una modificación una nueva versión es introducida. En este caso la información actualmente accesible en la base de datos, representaría el estado actual, la información que actualmente es válida, por lo que para recuperar el estado anterior se realizaría un proceso de rollback, que nos permita recuperar el estado de la información que fue actual en un momento anterior.

Al contrario que en el modelo Snapshot, no se registra toda la información nuevamente sino únicamente aquella entidad que ha sufrido los cambios. Esto permite solucionar el problema de la duplicación de la información. Además permite de esta forma reflejar expresamente la evolución puesto que los cambios son almacenados como diferencias con respecto a la versión anterior.

2.4.1.3 Modelo de datos basado en time-stamping

N. A. Lorentos, J. R. Rios Viqueira, N. Tryfona, presentan un modelo en el que el tiempo es considerado como absoluto discreto y lineal, en el que se permiten múltiples granularidades. En él, cada objeto espacial considerado se le asigna un tiempo de vida marcado por los dos extremos de un intervalo cerrado. En este modelo los tipos espaciales, punto, línea y región, tienen asociada una información temporal que permite registrar el tiempo de validez de esa información.

En este modelo, los cambios en los objetos son recogidos mediante la incorporación de aquella información que sufre el cambio, no de todos aquellos objetos entre los que

puede existir una relación de vecindad, dado que cada uno de ellos es considerado de forma independiente del resto.

No se hace una especificación explicita en el modelo de si el intervalo de tiempo que se registra es el tiempo de validez o de transacción, por lo que su aplicación podría depender de las necesidades de la aplicación para la que fuera emplearse. Asimismo, tampoco se hace ninguna restricción en cuanto a la granularidad temporal que podría aplicarse.

La ventaja que ofrece este modelo es que las operaciones pueden ser aplicadas uniformemente tanto a datos espaciales, como temporales o espaciotemporales.

Sin embargo, no recoge de una forma explícita la evolución respecto al movimiento de los objetos. Cada alteración sufrida por un objeto, generaría una nueva tupla en la que la relación con la versión anterior ha de ser reflejada por el usuario mediante la introducción de algún otro atributo, lo que la hace poco adecuada para este tipo de aplicaciones.

2.4.1.4 Modelos de datos espacio-temporal con objetos en movimiento

Una aproximación distinta al problema consiste en la incorporación al modelo de tipos de datos espacio-temporales. Se trata en este caso de contemplar un concepto conocido como objeto en movimiento.

Se trata de por tanto de aproximaciones que permiten gestionar tanto el cambio como el movimiento de los objetos, con un tratamiento totalmente general, dado que esta evolución puede contemplarse tanto si es continua como discreta. Si únicamente nos interesa registrar el movimiento del objeto, se puede contemplar el objeto como una abstracción en un punto y emplear un "moving point" para seguir su evolución es decir, podemos considerarlo como una función desde el tiempo en el plano bidimensional o como una polilínea en el espacio tridimensional. En cambio, si son cambios en la extensión del objeto, el empleo de las "moving region" nos permite registrar sus variaciones geométricas, tanto el crecimiento como la disminución del mismo. Se puede considerar esta última como un subconjunto del plano con un interior no vacío o como un polígono con agujeros, según la representación sea o no finita, respectivamente.

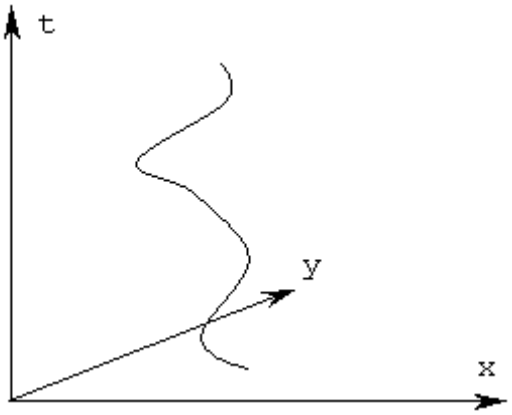

Figura 6: Moving point

2.4.1.5 Modelo de espacio-temporal Entidad-Relación

N. Tryfona y Th. Hadzilacos en 1998 definieron un modelo espacio-temporal que contempla el diseño de aplicaciones espacio-temporales, mediante una extensión al modelo entidad-relación y que es conocida como ST-ER. Definen un modelo para ser usado en la fase de diseño conceptual de cualquier aplicación espaciotemporal, previa a la fase de implementación.

El modelo ST-ER puede ser empleado para formalmente para:

- Definir los objetos, atributos y relaciones del dominio de aplicación.
- Expresar las vistas de usuario y las consultas al entorno de la base de datos.

Tras definir los objetos, como entidades del mundo real, caracterizados por un conjunto de atributos, de un determinado tipo de datos, e instancias de una determinada clase, cuyo comportamiento queda definido por un conjunto de métodos, introduce el concepto de los objetos espaciales. Para ello tienen en cuenta que para todo objeto en el mundo real su posición queda definida en función de su localización, forma, tamaño y orientación; siendo el modelo de espacio, empleado para definir la localización, euclídeo y definido como homomórfico a los reales. Con el fin de representar la forma se introducen tipos geométricos 0-dimensionales (puntos), 1-Dimensionales (líneas) y 2- Dimensionales (regiones). De esta manera las posiciones son definidas por una generalización (Figura 7).

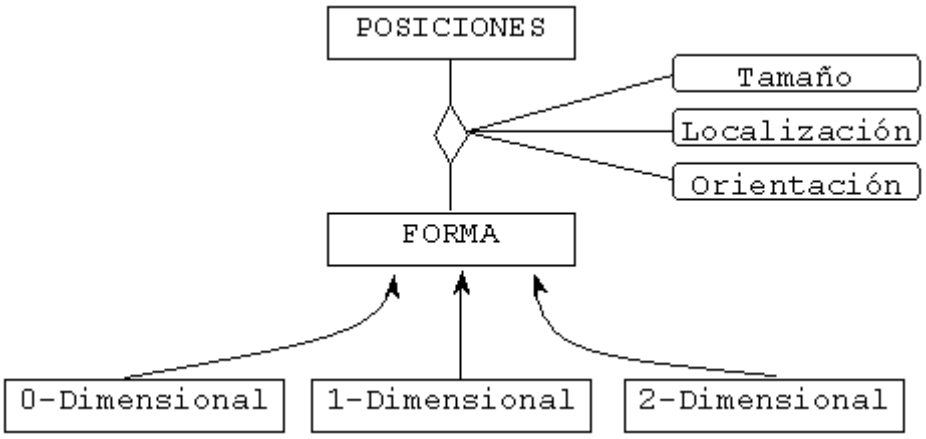

Figura 7: Modelo de entidades

La dimensión temporal en la que se basa el modelo es lineal, absoluta y discreta, en la que refleja tanto el tiempo de validez como de transacción, de manera que cualquier objeto, atributo y relación le puede ser asignada esa dimensión temporal, y puede por tanto ser indexado sobre ella. Sin embargo, no se considera que los objetos espaciales tengan una extensión temporal realizándose operaciones sobre ellos sin tener en cuenta posibles cambios futuros, es decir, el objeto geométrico es únicamente definido por su posición, por lo que si ésta sufre una modificación, el objeto en sí mismo cambia, se crea un nuevo objeto. Los atributos que varían en el espacio son propiedades del espacio, que indirectamente llegan a ser propiedades de los objetos situados en esa misma posición del espacio.

Un ejemplo de un posible ST-ER se muestra en la siguiente figura:

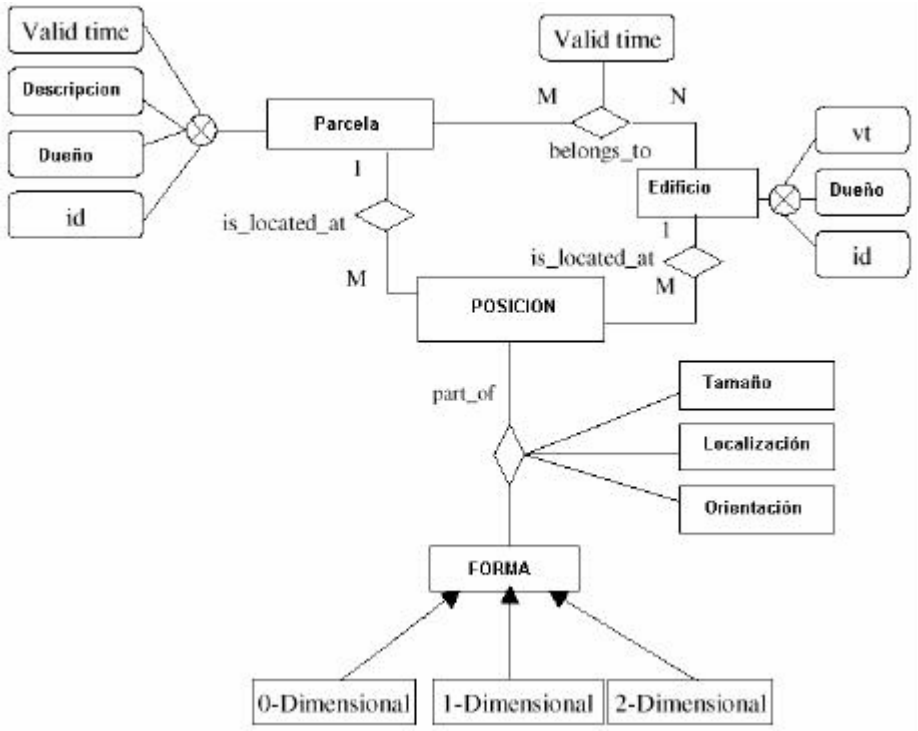

Figura 8: Ejemplo de ST-ER

En este caso se considera que los atributos poseen una variación, tanto en tiempo como en geometría, como propiedades del espacio, es decir, los atributos que sufren un cambio espacial a lo largo del tiempo, dependen únicamente de la posición no del objeto en sí mismo, cada nueva posición genera un nuevo objeto espacial, una nueva forma. Este tipo de variación se puede contemplar estableciendo una relación uno a muchos entre la entidad y las diferentes posiciones de ellas. En cambio, una relación puede tener diferentes versiones a lo largo del tiempo, por lo que aunque siga siendo la misma, las características que la describen podrían cambiar a lo largo del tiempo.

Se trata en este caso de proporcionar un medio de modelar aplicaciones espaciotemporales de manera que posteriormente puedan ser integradas dentro de una base de datos relacional. En la Figura 8, se modela el ejemplo de gestión catastral empleando ST-ER. La parcela puede tener asociadas diferentes posiciones a lo largo del tiempo, pero no son las posiciones las que tienen asociadas el tiempo de validez, sino que éste es aplicado sobre el objeto parcela como un atributo de ésta. No aparece en este caso el tiempo de transacción, por no considerarse necesario en el sistema pero podría ser incluido igual que el tiempo de validez.

Este modelo se desarrolló enfocado a Sistemas de Información Geográfica. En particular, a la gestión catastral sin un número de alteraciones tan elevado. En el modelo

no se incluyen referencias acerca de la granularidad del dominio espacial, ni ninguna aproximación de cómo realizar consultas sobre el modelo generado, ni operaciones más allá de las que proporciona el modelo relacional.

2.4.1.6 Modelo de datos espacio-temporal orientado a objetos

Los modelos de datos orientados a objetos están basados en el paradigma de la orientación a objetos que incluye objetos, clases, encapsulación, herencia y polimorfismo como herramientas. Diferentes aproximaciones pueden acometerse mediante el empleo de este concepto, para introducir la dimensión temporal en el diseño de aplicaciones espaciotemporales. Así, se embebe dentro de una misma entidad las diferentes versiones que ésta ha tendido a lo largo del tiempo, registrando su evolución. Otra aproximación sería construir un espacio 3-dimensional o 4–dimensional en la que el tiempo formará la última dimensión, que se presenta como la alternativa más intuitiva.

2.4.1.7 Modelos de datos espacio-temporales polinomiales

En 1998 S. Grumbach presentan un modelo de datos basado en el paradigma de las restricciones cuya idea básica es representar de forma finita colecciones de puntos en espacios d-dimensionales. La representación finita usa restricciones expresadas en un lenguaje de primer orden interpretado en algún dominio aritmético, como por ejemplo el de los racionales y las restricciones lineales $\leq$ y + sobre éste. En esta aproximación introduce el espacio y el tiempo como componentes naturales de conjuntos de puntos tridimensionales. Este diseño trata de reflejar el mismo equilibrio que al reflejar la información espacial, es decir, una limitada pérdida de corrección a favor de una mayor eficiencia computacional.

Ahora introduce un nuevo dominio, D, para tratar con esa nueva dimensión temporal de manera que las fórmulas son expresadas en términos de $L \cup D$, con dos tipos de restricciones:

- restricciones de igualdad sobre los objetos de D.
- restricciones lineales sobre los objetos de Q.

Se captura mediante este modelo la evolución temporal de la información espacial, pero a cambio presenta una criticada dificultad para poder establecer consultas de forma

sencilla. Ofrece limitaciones para expresar cálculos de distancia o predicados de conectividad. Evita los problemas presentados en los primeros modelos en cuanto a redundancia en el almacenamiento. Aunque no introduce ninguna referencia en cuanto a granularidad de la información temporal o la aplicabilidad del tiempo de validez o de transacción, dejando libertad al usuario.

3 OBJETIVOS

El principal objetivo del proyecto va a ser el aprendizaje de Oracle Spatial dentro de un dominio, por lo que lo primero que se debe hacer es buscar documentación sobre este tema, incluyendo instalación de Oracle 10g, configuración del producto y manejo de bases de datos espacio-temporales con Oracle Spatial.

El siguiente paso será diseñar la base de datos aplicando los conocimientos anteriores, utilizando los modelos de Entidad / Relación y Relacional. La complicación de estos diagramas consistirá en aplicar los datos espacio-temporales necesarios para realizar la aplicación.

A partir de estos modelos se pasará a crear la base de datos. Con la base de datos creada se diseñarán los disparadores necesarios para el correcto funcionamiento de la aplicación, así como el diseño de consultas necesarias para completar la funcionalidad.

Una vez terminados los pasos anteriores se procederá a diseñar la interfaz en Visual Basic 6.0. Para ello se deberán consultar manuales que ayuden a realizar la aplicación basada en este lenguaje. La interfaz se utilizará para consultar datos en momentos y espacios concretos partiendo de datos introducidos por el usuario. Para ello se utilizarán las consultas anteriormente mencionadas.

Durante este proceso se deberá generar de forma paralela la memoria explicando paso a paso los desarrollos y diseños realizados.

Con estos desarrollos se dará por finalizada la aplicación.

4 ENTORNO DE TRABAJO

4.1 Introducción a ORACLE SPATIAL

Un Sistema de Información Geográfica (SIG o GIS), es un tipo de Sistema de Información que, a través de una componente localizada en la superficie terrestre, proporciona acceso a información asociada a esa componente y viceversa. Se trata de un sistema para la creación y gestión de información espacial. En un sentido más estricto, es un sistema de información capaz de integrar, almacenar, editar, analizar y mostrar en pantalla información referenciada geográficamente. En un uso más genérico, un GIS es una aplicación con un "mapa inteligente" que permite que los usuarios creen consultas interactivas (búsquedas creadas por el usuario), analicen información espacial y editen datos.

Existen aplicaciones especializadas que son capaces de extraer la información contenida en los GIS y presentarla de manera grafica como es el caso de Oracle Spatial.

La utilidad principal de un Sistema de Información Geográfica radica en su capacidad para construir modelos o representaciones del mundo real a partir bases de datos espaciales.

Algunas de sus aplicaciones principales son:

- ✓ Cartografía automatizada: Construcción y mantenimiento de planos digitales de cartografía.
- ✓ Infraestructura: Almacenamiento de información alfanumérica de servicios relacionados con las distintas representaciones gráficas de los mismos para el desarrollo, mantenimiento y administración de redes de electricidad, gas, agua, teléfono, alcantarillado, etc.
- ✓ Gestión territorial: Creación de aplicaciones SIG dirigidas a la gestión de entidades territoriales.
- ✓ Medio ambiente: Sistemas SIG que facilitan la evaluación del impacto ambiental en la ejecución de proyectos.
- ✓ Equipamiento social: Implementación de aplicaciones GIS dirigidas a la gestión de servicios de impacto social, tales como servicios sanitarios, centros escolares.

Oracle Spatial proporciona un esquema SQL de las funciones que facilitan el almacenamiento, la recuperación, la actualización y las consultas. Las componentes de estas funciones son las siguientes:

- Un esquema (MDSYS) que prescribe el almacenaje, la sintaxis, y la semántica de los tipos de datos geométricos apoyados.
- Un mecanismo de indización espacial.
- Un sistema operador y las funciones para realizar queries de la sección de interés, espaciales join queries.
- Utilidades administrativas.

Oracle Spatial está apoyado en el modelo objeto-relacional con el fin de la representación geométrica de una característica espacial (geometría).
El modelo de datos de Oracle Spatial es una jerarquía consistente en:

- Elementos: Un elemento es la unidad de información básica de la geometría. Los tipos espaciales apoyados del elemento son puntos, líneas, y polígonos.
- Geometrías: Es la representación de una característica espacial, modelada como conjunto ordenado de elementos primitivos. Puede consistir en un solo elemento o una colección (homogénea o heterogénea) de elementos.
- Capas: Una capa es una colección de geometrías que tienen el mismo conjunto de atributos.

Los beneficios de utilizar Oracle Spatial se pueden resumir en los siguientes puntos:

- Elimina la necesidad de dobles arquitecturas, ya que todos los datos se pueden almacenar de la misma forma. Un almacenamiento unificado significa que todos los tipos de datos (textos, mapas, multimedia) están almacenados juntos., en vez estar almacenados por separado.
- Utiliza SQL, un lenguaje estándar para acceder a bases de datos relacionales, de esta manera elimina la necesidad de un lenguaje específico para manejar datos espaciales.

- Define el tipo de datos SDO_GEOMETRY, el cual es esencialmente equivalente a los tipos de datos espaciales de los estandars OGC y SQL / MM.
- Elimina la necesidad de organizaciones separadas para el mantenimiento de una infraestructura de datos espaciales (HW, SW, soporte, etc) y elimina la necesidad de herramientas específicas y habilidades para operar con datos espaciales.

4.2 Manejo de la información espacial

La geocodificación de los tipos de datos espaciales (Geocoding), consiste en traducir una dirección cualquiera en coordenadas x, y.
Las operaciones espaciales normalmente incluyen los siguientes puntos:

- Almacenamiento de datos espaciales. En la mayoría de los casos implica los siguientes puntos:
  - Almacenar los datos de la forma apropiada en la base de datos. El tipo de datos utilizado para el almacenamiento es del tipo geométrico (puntos, líneas, polígonos, etc.).
  - Insertar, borrar y actualizar los tipos de datos espaciales en la base de datos.

- Análisis del vector de datos espaciales. Normalmente incluye los siguientes análisis de funcionalidad.
  - Within_distance: identifica todos los datos espaciales dentro de una distancia específica a partir de la localización consultada.
  - Contains: identifica todos los datos espaciales que contienen una localización específica consultada.
  - Nearest_neighbor: identifica todos los datos espaciales más cercanos a una localización consultada.
  - Distance: calcula la distancia entre dos objetos espaciales.
  - Buffer: construye zonas de amortiguamiento o separación alrededor de datos espaciales.

- Overlay: Solapa o recubre diferentes capas de los datos espaciales.
- Visualization: Presenta datos espaciales usando mapas.

- Análisis de datos de red. La mayoría de los datos espaciales se pueden también representar como una red de datos.

Por otro lado, el almacenamiento de los datos espaciales se representa mediante los siguientes tipos de datos básicos:

- Puntos: requieren sólo las coordenadas x, y. También puede encontrarse la z si se trata de tres dimensiones.
- Líneas: se necesita una coordenada de inicio, una coordenada de fin y además una serie de coordenadas intermedias.
- Polígonos o regiones: son líneas cerradas.

4.3 Visión general del ORACLE SPATIAL

4.3.1 Modelo de datos

El modelo de datos es la forma en que se almacenan los datos espaciales. Para especificar dicha información se utilizan dos componentes:

- ✓ Localización específica: el dato se ubica respecto a dos, tres o cuatro dimensiones.
- ✓ Formas específicas de las estructuras geométricas de los datos: las posibles formas pueden ser punto, líneas o polígonos.

El tipo de datos SDO_GEOMETRY captura la información sobre localización y forma de los datos almacenados en las filas de una tabla. Este tipo de datos se representa internamente en Oracle como un objeto. A su vez, se utiliza para crear columnas donde almacenar la localización de objetos.

4.3.2 Consulta y análisis

La consulta y el análisis proporcionan la funcionalidad principal para analizar y consultar geometrías espaciales. Se distinguen dos componentes principales: Geometry Engine e Index Engine. Es a través de estos componentes que se puede llevar a cabo consultas y análisis espaciales.

El componente Geometry Engine proporciona funciones para analizar, comparar y manipular geometrías.

El componente Index Engine se utiliza para acelerar las consultas en una tabla a través de la creación de índices.

4.4 Localización y habilitación de las aplicaciones

4.4.1 Añadir información de localización en tablas

La mayoría de los datos de las aplicaciones se encuentran dentro de dos tipos de tablas. Por un lado las "Tablas de aplicaciones específicas" contienen toda la información específica para la aplicación. Las tablas de aplicación utilizan técnicas estándares de normalización para llegar a las tablas apropiadas donde almacenar los datos. Por otro lado están las "Tablas geográficas" en las cuales los datos geográficos son independientes de la aplicación y contienen columnas para almacenar información espacial explícita. Estos datos pueden ser utilizados como un valor añadido en la aplicación.

4.4.2 Consideraciones para diseñar una aplicación con datos específicos

La organización de una aplicación con datos específicos en las correspondientes tablas, será una aplicación dependiente y probablemente se utilizarán técnicas de diseño estándares como normalización, entidad-relación (ER), etc. En cambio, Oracle Spatial no posee ninguna recomendación específica o restricción para el modo en que se organizan los datos.

Un punto importante a considerar son las particiones cuando las tablas de datos poseen millones de filas.

4.4.3 Datos geográficos

Cada dato geográfico está normalmente disponible en una variedad de fuentes y se deben almacenar en una Base de Datos. Los datos geográficos constan de los siguientes atributos:

- Estados: incluye el nombre del estado, la abreviatura, la población, la media de ingresos familiar y el límite (almacenado en un objeto SDO_GEOMETRY).

- Condados: incluye el nombre del condado, el nombre del estado al cual pertenece, el área, la población por milla cuadrada y un objeto SDO_GEOMETRY para almacenar el límite del condado.

- Interestados: incluye el nombre y un objeto SDO_GEOMETRY para almacenar la forma lineal.

- Calles: incluye el nombre, ciudad, estado y un objeto SDO_GEOMETRY para almacenar la forma lineal de la calle.

Almacenar las calles, los interestados, los condados y estados en una sola tabla es ineficiente y debe ser evitado. Se deben almacenar los datos en diferentes tablas basadas en los siguientes criterios generales:

- Separar los datos espaciales que no comparten los mismos atributos. Es similar a las técnicas de normalización utilizadas para regular datos.

- Separar los datos robustos de los datos simples. Si se tienen dos tipos de datos, y el numero de apariciones de uno es mucho mayor que el del otro, el almacenar los dos tipos de datos en la misma tabla puede ocasionar problemas en la realización cuando solo se quiere acceder al dato mas simple.

- Separar basándose en la forma geométrica. Si se separa según sea un punto, una línea o un polígono, entonces se pueden utilizar las

herramientas proporcionadas por Oracle Spatial para comprobar el tipo de geometría a la hora de insertar.

- Particionar datos localizados. En el caso de almacenar un tipo de dato en una tabla con un gran número de apariciones, es muy útil particionar estos datos.

4.4.4 Metadatos para tablas espaciales

A todos los objetos geométricos SDO_GEOMETRY que se encuentran en una determinada columna de una tabla, Oracle Spatial los define como capas espaciales. Debido a esto, es necesario especificar un determinado metadato para cada capa. Se incluye la siguiente información:

- El número de dimensiones
- Los límites de cada dimensión
- La tolerancia para cada dimensión
- El sistema de coordenadas

Esta información para cada capa espacial se almacena en el diccionario USER_SDO_GEOM_METADATA.

4.4.4.1 Diccionario para Metadatos espaciales

El diccionario está formado por los siguientes atributos:

- Nombre_Tabla y Nombre_Columna: ambos identifican unívocamente cada capa espacial.
- Diminfo: almacena información sobre las dimensiones de una capa, con el fin de identificar dicha capa.
- Srid: almacena información sobre el sistema de coordenadas correspondiente al dato geométrico.

El atributo DIMINFO es del tipo SDO_DIM_ARRAY. Este tipo de datos es de longitud de array, por lo que cada elemento tiene el tamaño acorde al número de dimensiones.

Por lo tanto si es una geometría bidimensional el atributo DIMINFO contendrá dos tipos SDO_DIM_ELEM.

Cada tipo SDO_DIM_ELEM almacena información sobre una dimensión específica y contiene los siguientes campos:

- ✓ SDO_DIMNAME: almacena el nombre de la dimensión (puede ser uno para la longitud y otro para la latitud).
  - SDO_LB y SDO_UB: estos dos números definen el límite inferior y el límite superior de una dimensión específica.
  - SDO_TOLERANCE: es un valor de tolerancia, utilizado para especificar un grado de precisión del dato espacial.

Dos puntos de una geometría no se pueden considerar duplicados aunque su distancia sea menos que la tolerancia.

La tolerancia debería ser la distancia más pequeña distinguida en tu aplicación, aunque también puede ser la media entre la diferencia de dos coordenadas.

Por otro lado, en el atributo SRID el sistema de coordenadas se puede encontrar en uno de los siguientes:

- Geodetic: coordenadas angulares, expresadas en términos de longitud, latitud con respecto a la superficie de la Tierra.
- Proyected: coordenadas cartesianas que resultan de un mapa matemático realizado desde un área de la superficie de la Tierra a un plano.
- Local: sistema de coordenadas cartesianas sin conexión con la superficie de la Tierra y a veces específicas de la aplicación.

4.5 Tipo de datos SDO_GEOMETRY

La descripción geométrica de un objeto espacial se almacena en una sola fila, en una sola columna del tipo SDO_GEOMETRY del objeto en una tabla definida por el usario. Cualquier tabla que tenga una columna del tipo SDO_GEOMETRY debe tener otra columna, o fijado de columnas, que define una clave primaria única para esa tabla. Las tablas de esta clase se denominan "Tablas espaciales" o "Tablas espaciales de la geometría".

4.5.1 Tipos de geometrías espaciales en ORACLE

En Oracle existen distintos tipos de geometrías .En el apartado 3.2 se han explicado las mas importantes, pero existen diversas ramificaciones de dichos tipos de geometrías. Las distintas geometrías de Oracle Spatial son las siguientes:

- Puntos y regiones de punto
- Secuencias de líneas
- Polígonos de n-punto
- Arcos
- Polígonos
- Polígonos compuestos
- Líneas compuestas
- Círculos
- Rectángulos optimizados

Los puntos de dos dimensiones son elementos integrados por dos ordenadas, X e Y, a menudo correspondiendo a la longitud y a la latitud. La secuencia de líneas se compone de uno o más pares de puntos que definen el segmento de línea. Los polígonos se componen de secuencias de líneas que forman un anillo cerrado, y el área del polígono se implica. Por ejemplo, un punto puedo representar una localización de un edificio, una secuencia de líneas puedo representar un camino o una trayectoria de vuelo, y un polígono puedo representar un estado, una ciudad, un distrito o un bloque de una ciudad.

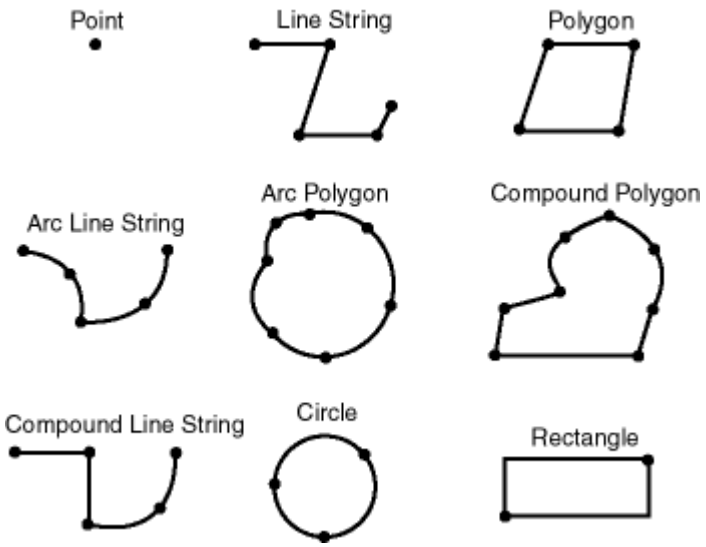

Figura 9: Tipos de geometrías

Existe además otro tipo de geometría denominada Colección. Esta formada por múltiples elementos geométricos.

4.5.2 Implementación del tipo de datos SDO_GEOMETRY

El tipo de datos SDO_GEOMETRY se compone de dos componentes lógicos. Uno de ellos es el Sistema de Referencia Espacial, también llamado Sistema de Coordenadas. El otro componente es un array de elementos (ElementArray), el cuál describe la forma y localización del tipo de datos SDO_GEOMETRY (con referencias al sistema de coordenadas). Este array constituye el objeto de SDO_GEOMETRY. Representa cualquiera de los tipos de datos espaciales (puntos, líneas, polígonos, etc).

A su vez, el tipo de datos SDO_GEOMETRY esta compuesto de los siguientes atributos:

- SDO_GTYPE: número que se utiliza para especificar el tipo de forma geométrica representada, pero no sus coordenadas.
- SDO_SRID: número que especifica el sistema de referencia espacial donde se encuentra la localización.
- SDO_POINT: atributo que indica las coordenadas de un punto.
- SDO_ELEM_INFO: array que especifica dentro del array SDO_ORDINATES dónde la posición en la comienza un nuevo elemento, cómo está conectado (líneas rectas o arcos) y si es un punto, una línea o un polígono.

- SDO_ORDINATES: array que almacena las coordenadas de todos los elementos geométricos.

Si la geometría en cuestión es una forma arbitraria, se almacenan las coordenadas utilizando los atributos SDO_ORDINATES y SDO_ELEM_INFO del array. En caso contrario estos atributos tendrían valor NULL.
La forma de declaración del tipo SDO_GEOMETRY según Oracle Spatial es el siguiente:

```
CREATE TYPE SDO_GEOMETRY AS OBJECT (
SDO_GTYPENUMBER,
SDO_SRIDNUMBER,
SDO_POINTSDO_POINT_TYPE,
SDO_ELEM_INFOSDO_ELEM_INFO_ARRAY,
SDO_ORDINATESSDO_ORDINATE_ARRAY);
```

A su vez, también es necesaria la declaración de los tipos de datos SDO_POINT_TYPE, SDO_ELEM_INFO_ARRAY y SDO_ORDINATE_ARRAY:

```
        CREATE TYPE SDO_POINT_TYPE AS OBJECT (
XNUMBER,
YNUMBER,
Z,NUMBER);
        CREATE TYPE SDO_ELEM_INFO_ARRAY AS VARRAY (1048576) OF NUMBER;
CREATE TYPE_SDO_ORDINATE_ARRAY AS VARRAY (1048576) OF NUMBER;
```

A continuación se procederá a explicar cada atributo de forma más detallada.

4.5.2.1 Atributo SDO_GTYPE

El atributo SDO_GTYPE describe el objeto geométrico a un alto nivel. Esto significa que el objeto geométrico puede ser una combinación de múltiples elementos, cada uno con una forma diferente, pero el atributo SDO_GTYPE especifica la forma general para el objeto entero, es decir, todos sus componentes. Es por esto que este atributo puede tomar valores distintos dependiendo de si se trata de un punto, una línea, un polígono, una región de puntos, etc.
El valor de SDO_GTYPE es de 4 dígitos (DLTT) donde:

- "D" identifica el número de las dimensiones (2,3 ó 4) por lo que se pueden almacenar objetos espaciales bidimensionales, tridimensionales o cuatridimensionales.
- "L" identifica la dimensión lineal de la medida para el sistema de referencia para una geometría referenciada a un sistema de referencia, por lo cual la dimensión (3 ó 4) contendrá la dimensión de la medida. Para una geometría no referenciada a un sistema de referencia, Oracle Spatial acepta por defecto la última dimensión como la medida de la geometría, en caso contrario, si no tuviera la dimensión de la medida, sería 0.
- "TT" identifica el tipo de la geometría que representa el objeto. Los valores posibles se presentan en la siguiente tabla:

Valor TT	Tipo de geometría	Descripción
DL00	Geometría desconocida	Oracle Spatial ignora esta geometría
DL01	Punto	La geometría contiene un punto
DL02	Línea o curva	La geometría contiene una secuencia de líneas que puede contener segmentos rectos o circulares
DL03	Polígono	La geometría contiene un polígono con o sin agujeros
DL04	Colección	La geometría es una colección heterogénea de elementos
DL05	Múltiples puntos	La geometría contiene uno o más puntos
DL06	Multilínea o Multicurva	La geometría contiene una o más secuencias de líneas
DL07	Multipolígono	La geometría puede contener múltiples polígonos

Tabla 1: Valores válidos de "TT" en SDO_GTYPE

4.5.2.2 Atributo SDO_SRID

El atributo SDO_SRID especifica el sistema de coordenadas para la geometría, como puede ser cartesiana (x, y) o terrestres (latitud, altitud). Si SDO_SRID es nulo, no se asocia ningún sistema coordinado a la geometría. Si SDO_SRID no es nulo, debe contener un valor de la columna SRID de la tabla de SDO_COORD_REF_SYS, y, a su

vez, este valor se debe insertar en la columna de SRID de la tabla USER_SDO_GEOM_METADATA.

Los sistemas de referencia se almacenan en la tabla MDSYS.CS_SRS. Estos sistemas se dividen en geodéticos, proyectados y locales. El sistema geodético está basado en coordenadas geográficas, las cuales son coordenadas angulares (longitud y latitud). El sistema proyectado está basado en las coordenadas cartesianas que indican las posiciones referentes sobre una determinada zona. Por último, el sistema local se basa también en coordenadas cartesianas dentro de un sistema coordinado de la no-Tierra (no-georeferencial).

Es necesario que todas la geometrías pertenecientes a la misma columna de una tabla deben tener el mismo sistema de coordenadas. Es decir, el mismo valor de SDO_SRID.

4.5.2.3 Atributo SDO_POINT

El atributo SDO_POINT especifica la localización de un punto geométrico. Para su definición se utiliza el tipo de objeto SDO_POINT_TYPE, el cual contiene tres atributos X, Y, Z de tipo numérico. Si los atributos de SDO_ELEM_INFO y de SDO_ORDINATES son nulos, y a su vez, el atributo SDO_POINT no es nulo, entonces los valores de X e Y se consideran las coordenadas de una geometría de tipo punto. En caso contrario, el atributo de SDO_POINT es ignorado por Oracle Spatial. Con el fin de alcanzar un almacenamiento óptimo, se deben almacenar las geometrías de tipo punto en el atributo SDO_POINT.

4.5.2.4 Atributo SDO_ELEM_INFO

El atributo SDO_ELEM_INFO es un array compuesto por tres elementos todo ellos numéricos. Esto quiere decir que el tamaño del array siempre va a ser un multiplo de 3. Además, cada terna está asociada con un elemento de la geometría. Por otro lado, este atributo nos permite saber cómo interpretar las órdenes guardadas en el atributo SDO_ORDINATES.

La estructura de la terna que contiene es la siguiente:

<offset, element-type, interpretation>

- SDO_STARTING_OFFSET: Indica el índice donde se encuentra almacenada la primera ordenada en el array de SDO_ORDINATES. El

valor mínimo permitido es 1, por lo que la primera ordenada para el primer elemento sería SDO_GEOMTRY.SDO_ORDINATES(1).

o SDO_ETYPE y SDO_INTERPRETATION: Indican el tipo de elemento de la geometría. Los valores posibles de estos elementos se muestran en la siguiente tabla:

Sdo_Etype	Sdo_Interpretation	Significado
0	Cualquier valor numérico	Geometría no soportada por Oracle Spatial.
1	1	Geometría de tipo Punto.
1	9	Geometría orientada a tipo Punto.
1	n>1	Conjunto de n puntos.
2	1	Line Strings (conectores) cuyos vértices son conectados por segmentos de líneas rectas.
2	2	Conectores cuyos vértices son conectados por arcos.
1003 ó 2003	1	Polígono simple cuyos vértices son conectados por líneas rectas. Se debe especificar un punto para cada vértice y el último punto especificado debe ser exactamente igual al primero para cerrar el polígono.
1003 ó 2003	2	Polígono simple cuyos vértices son conectados por arcos.
1003 ó 2003	3	Geometría de tipo rectángulo, a veces llamado rectángulo optimizado. Definido únicamente a partir de dos puntos que son el del límite inferior izquierdo y el del límite superior derecho.
1003 ó 2003	4	Geometría de tipo círculo. Definido a partir de tres puntos no lineales del arco de la circunferencia.

4	n>1	Geometría de tipo Line String. Algunos de sus vértices son conectados por líneas rectas y otros por arcos.
1005 ó 2005	n>1	Polígonos compuestos. Algunos de sus vértices son conectados por líneas rectas y otros por arcos.

Tabla 2 : Valores y semántica de SDO_ELEM_INFO

4.5.2.5 Atributo SDO_ORDINATES

El atributo SDO_ORDINATES almacena las coordenadas de cualquier dimensión de todos los elementos de la geometría. Se define mediante un array de longitud variable de valores numéricos que almacenan los valores de las coordenadas para configurar la geometría. Este array debe ser siempre usado en conjunción con el array de SDO_ELEM_INFO. Los valores en el array son ordenados por dimensiones. Por ejemplo, un polígono cuyo límite tiene cuatro puntos bidimensionales es almacenado como {X1, Y1, X2, Y2, X3, Y3, X4, Y4, X1, Y1}. Todos los valores en el array deben ser válidos y no nulos.

4.5.3 Ejemplos de geometrías SDO_GEOMETRY válidas

Hay que mencionar que una vez definidos todos los componentes del objeto SDO_GEOMETRY se valida que el objeto definido resulta una geometría válida. Si no fuera una geometría válida no se podría utilizar dicha geometría hasta no validarla correctamente.
A continuación se mostraran unos ejemplos de geometrías SDO_GEOMETRY.

- Punto

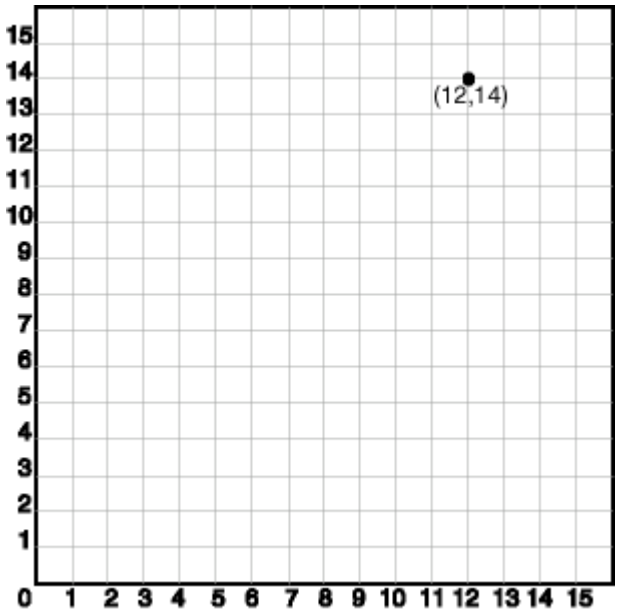

Figura 10: Ejemplo de geometría Punto

La definición de la geometría Punto mediante Oracle Spatial sería la siguiente:

	Geometría Punto
SDO_GTYPE	2001
SDO_SRID	NULL
SDO_POINT	(12,14,NULL)
SDO_ELEM_INFO	NULL
SDO_ORDINATES	NULL

Tabla 3: Definición de geometría Punto

- Secuencia de líneas

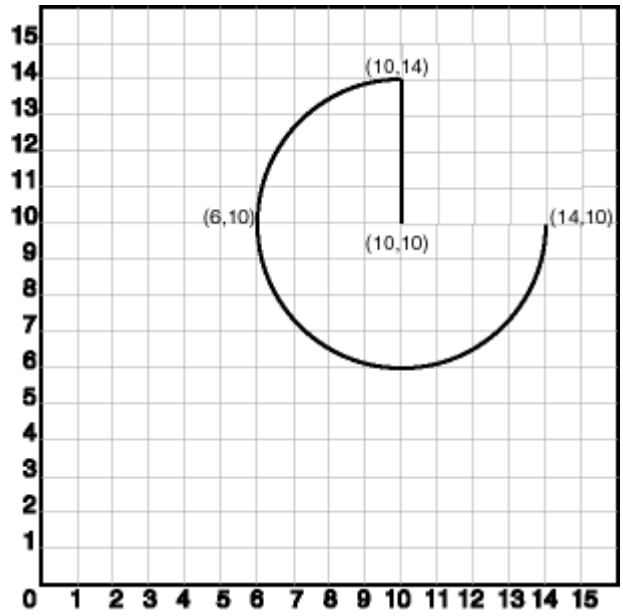

Figura 11: Ejemplo de geometría Secuencia de Líneas

La definición de la geometría Secuencia de Líneas mediante Oracle Spatial sería la siguiente:

	Geometría Secuencia de Líneas
SDO_GTYPE	2002
SDO_SRID	NULL
SDO_POINT	NULL
SDO_ELEM_INFO	(1,4,2,1,2,1,3,2,2)
SDO_ORDINATES	(10,10,10,14,6,10,14,10)

Tabla 4: Definición de geometría Secuencia de Líneas

- Rectángulo

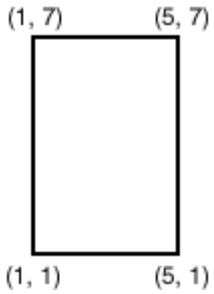

Figura 12: Ejemplo de geometría Rectángulo

La definición de la geometría Rectángulo mediante Oracle Spatial sería la siguiente:

	Geometría Rectángulo
SDO_GTYPE	2003
SDO_SRID	NULL
SDO_POINT	NULL
SDO_ELEM_INFO	(1,1003,3)
SDO_ORDINATES	(1,1,5,7)

Tabla 5: Definición de geometría Rectángulo

- Polígono compuesto

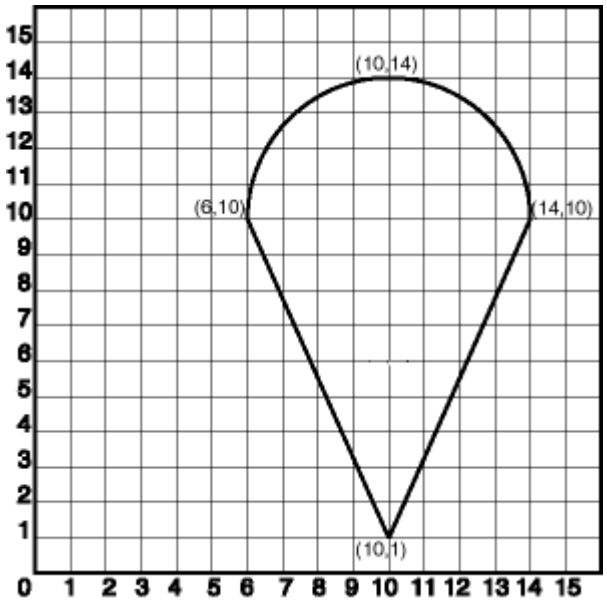

Figura 13: Ejemplo de geometría Polígono compuesto

La definición de la geometría Polígono compuesto mediante Oracle Spatial sería la siguiente:

	Geometría Polígono compuesto
SDO_GTYPE	2003
SDO_SRID	NULL
SDO_POINT	NULL
SDO_ELEM_INFO	(1,1005,2, 1,2,1, 5,2,2)
SDO_ORDINATES	(6,10, 10,1, 14,10, 10,14, 6,10)

Tabla 6: Definición de geometría Polígono compuesto

- Polígono con agujero

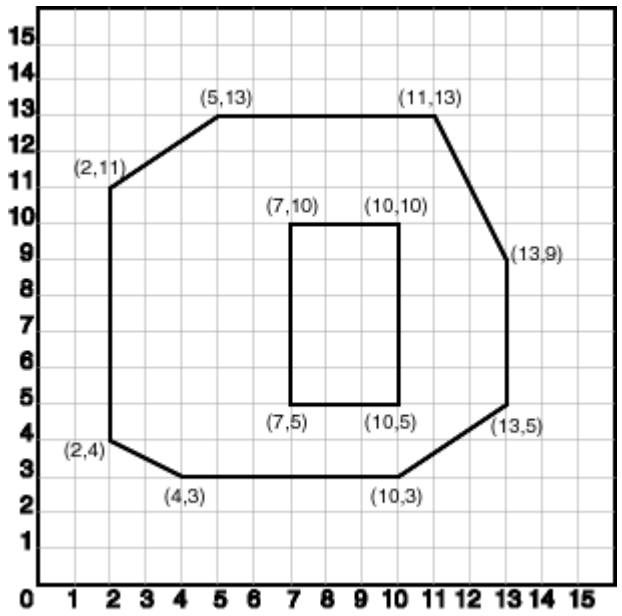

Figura 14: Ejemplo de geometría Polígono con agujero

La definición de la geometría Polígono con agujero mediante Oracle Spatial sería la siguiente:

	Geometría Polígono con agujero
SDO_GTYPE	2003
SDO_SRID	NULL
SDO_POINT	NULL
SDO_ELEM_INFO	(1,1003,1,19,2003,1)
SDO_ORDINATES	(2,4, 4,3, 10,3, 13,5, 13,9, 11,13, 5,13, 2,11, 2,4, 7,5, 7,10, 10,10, 10,5, 7,5)

Tabla 7: Definición de geometría Polígono con agujero

4.6 Visualización de datos espaciales

La tecnología de Oracle incluye el componente Map Viewer con el fin de facilitar la creación de mapas basados en datos espaciales. Cada mapa está asociado con un conjunto de temas. Cada tema a su vez, indica los datos espaciales de una tabla específica y está asociado a un "Estilo de presentación".

Oracle Spatial provee de un apropiado diccionario de vistas, USER_SDO_MAPS, USER_SDO_THEMES y USER_SDO_STYLES con el fin de definir nuevos mapas, asociarlos con temas y especificar interpretaciones de estilos para los temas internos en la base de datos, respectivamente.

Map Viewer por lo tanto, es una herramienta de visualización basada en java que usa información de localización de Oralce Spatial para construir y mostrar mapas en una búsqueda o en un contexto de aplicación específica. MapViewer se puede usar para:

- Crear mapas adaptados al cliente que muestren características tales como carreteras, áreas urbanas, tuberías y otras redes de transportes.
- Mostrar en mapas elementos frontera de la nación, de las comunidades o locales
- Visualizar datos de negocio (población demográfica, volúmenes de ventas...) para retratar y explorar relaciones que pueden expresarse mejor gráficamente como mapas geográficos

- Complementar el flujo de trabajo de las aplicaciones, proporcionando interacción con los datos de los mapas.

El Map Viewer fue desarrollado para simplificar la creación de aplicaciones que interpretan y presentan datos de localización como parte de aplicaciones de negocio en Internet e inalámbricas. Es posible más control en la imagen e interacción de la aplicación con el mapa a través de clientes Java APIs que soportan interacción en los mapas como zoom, localización y el recentrado.

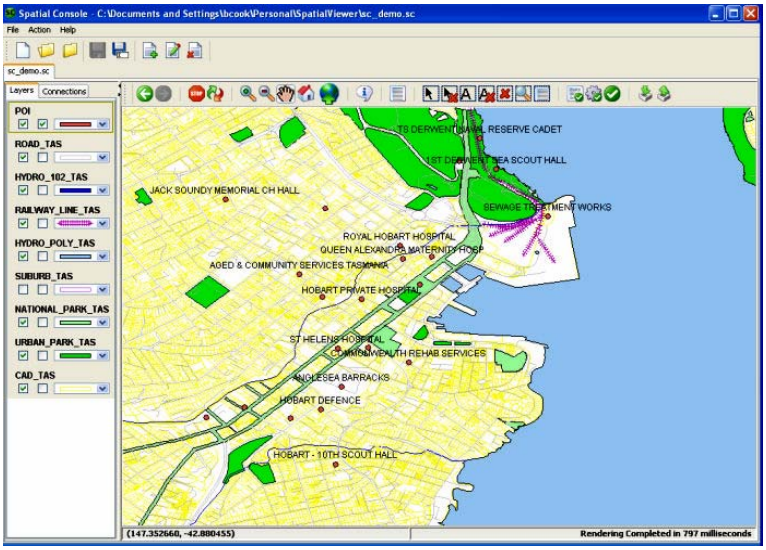

Figura 15: Ejemplo de Map Viewer

4.7 Visual Basic 6.0

4.7.1 Antecedentes históricos

El lenguaje de programación BASIC (Beginner's All purpose Symbolic Instruction Code) nació en el año 1964 como una herramienta destinado a principiantes, buscando una forma sencilla de realizar programas, empleando un lenguaje casi igual al usado en la vida ordinaria (en inglés), y con instrucciones muy sencillas y escasas. Teniendo en cuenta el año de su nacimiento, este lenguaje cubría casi todas las necesidades para la ejecución de programas.

Los autores fueron los científicos John G. Kemeny (Budapest, 1926 – USA 1992) y Thomas E. Kurtz (Illinois 1928) Su trabajo original se llamó True BASIC**.**

La evolución del BASIC por los años 70 fue escasa, dado el auge que tomaron en aquella época lenguajes de alto nivel como el FORTRAN y el COBOL. En 1978 se

definió una norma para unificar los Basics existentes creándose la normativa BASIC STANDARD.

Con la aparición de los primeros ordenadores personales, dedicados comercialmente al usuario particular, hacia la primera mitad de los 80, el BASIC resurgió como lenguaje de programación pensado para principiantes, y muchos de estos pequeños ordenadores domésticos lo usaban como único sistema operativo (Sinclair, Spectrum, Amstrad)
Con la popularización del PC, salieron varias versiones del BASIC que funcionaban en este tipo de ordenadores (Versiones BASICA, GW-BASIC), pero todas estas versiones del BASIC no hicieron otra cosa que terminar de rematar este lenguaje. Los programadores profesionales no llegaron a utilizarlo, debido a las desventajas de este lenguaje respecto a otras herramientas (PASCAL, C, CLIPPER). El BASIC con estas versiones para PC llegó incluso a perder crédito entre los profesionales de la informática. Las razones para ello eran obvias:

- ✓ No era un lenguaje estructurado.
- ✓ No existían herramientas de compilación fiables.
- ✓ No disponía de herramientas de intercambio de información.
- ✓ No tenía librerías.
- ✓ No se podía acceder al interior de la máquina.
- ✓ Una larga lista de desventajas respecto a otros lenguajes de programación.

Tal fue ese abandono por parte de los usuarios, que la aparición del Quick-BASIC de Microsoft, una versión ya potente del BASIC, que corregía casi todos los defectos de las versiones pasó prácticamente inadvertida, a no ser porque las últimas versiones del sistema operativo MS-DOS incluían una versión de Quick-BASIC algo recortada (Q-Basic) como un producto más dentro de la amplia gama de ficheros ejecutables que acompañan al sistema operativo, y aprovecha de él el editor de textos.
Esta versión del popular BASIC ya es un lenguaje estructurado, lo que permite crear programas modularmente, mediante subrutinas y módulos, capaz de crear programas ya competitivos con otros lenguajes de alto nivel. Sin embargo llegaba tarde, pues los entornos MS-DOS estaban ya superados por el entorno gráfico Windows.

Sin embargo algo había en el BASIC que tentaba a superarse: su gran sencillez de manejo. Si a esto se le añade el entorno gráfico Windows, el aprovechamiento al máximo de las posibilidades de Windows en cuanto a intercambio de información, de sus librerías, de sus drivers y controladores, manejo de bases de datos, etc. el producto resultante puede ser algo que satisfaga todas las necesidades de programación en el entorno Windows. La suma de todas estas cosas es VISUAL-BASIC. Esta herramienta conserva del BASIC de los años 80 únicamente su nombre y su sencillez, y tras su lanzamiento al mercado, la aceptación a nivel profesional hizo borrar por fin el "mal nombre" asociado a la palabra BASIC.

En el año 2001 se comercializó la versión 6.0 de este producto. Desde su salida al mercado, cada versión supera y mejora la anterior. Dados los buenos resultados a nivel profesional de este producto, y el apoyo prestado por el fabricante para la formación de programadores, Visual-Basic se ha convertido en la primera herramienta de desarrollo de aplicaciones en entorno Windows.

Es obligado decir sin embargo, que sigue siendo BASIC. No se pueden comparar sus prestaciones con otros lenguajes cuando deseamos llegar al fondo de la máquina y controlar uno a uno sus registros. No es ese el fin perseguido con VB y si es necesario llegar a esas precisiones será necesario utilizar otro lenguaje que permita bajar el nivel de programación. (Visual-C) o realizar librerías (DLLs) que lo hagan. En la mayor parte de las aplicaciones, las herramientas aportadas por VB son mas que suficiente para lograr un programa fácil de realizar y de altas prestaciones.

4.7.2 Características Generales de Visual-Basic

Visual-Basic es una herramienta de diseño de aplicaciones para Windows, en la que estas se desarrollan en una gran parte a partir del diseño de una interface gráfica. En una aplicación Visual Basic, el programa está formado por una parte de código puro, y otras partes asociadas a los objetos que forman la interface gráfica.

Es por tanto un término medio entre la programación tradicional, formada por una sucesión lineal de código estructurado, y la programación orientada a objetos. Combina ambas tendencias. Ya que no podemos decir que VB pertenezca por completo a uno de esos dos tipos de programación, debemos inventar una palabra que la defina Programación Visual.

La creación de un programa bajo Visual Basic lleva los siguientes pasos:

a. **Análisis.** Es el estudio de las necesidades que han dado origen a la creación de ese programa. Es lo que se llama *Análisis de la aplicación*. Es la primera fase que debe tener siempre un programa y es también la más olvidada entre los programadores noveles. Una aplicación no se inicia con el teclado, sino sobre un papel.

b. **Creación de un interfaz de usuario**. Este interfaz será la principal vía de comunicación hombre-máquina, tanto para salida de datos como para entrada. Será necesario partir de una o varias ventanas (formularios) a las que se le irán añadiendo los controles necesarios.

c. **Definición de las propiedades de los controles**. Se dará la forma, posición, y todas las características necesarias a los controles que se hayan colocado en el formulario. Estas propiedades determinarán la forma estática de los controles, es decir, como son los controles y para qué sirven.

d. **Generación del código asociado a los eventos que ocurran a estos controles.** A la respuesta a estos eventos (click, doble click, una tecla pulsada, etc.) se le llama Procedimiento, y deberá generarse de acuerdo a las necesidades del programa.

e. **Generación del código del programa**. Un programa puede hacerse solamente con la programación de los distintos procedimientos que acompañan a cada objeto. Sin embargo, VB ofrece la posibilidad de establecer un código de programa separado de estos eventos. Este código puede introducirse en unos bloques llamados Módulos, en otros bloques llamados Funciones, y otros llamados Procedimientos. Estos Procedimientos no responden a un evento acaecido a un control o formulario, sino que responden a un evento producido durante la ejecución del programa.

*4.8 Oracle SQL*Plus*

La herramienta que nos proporciona ORACLE para interactuar con la base de datos se llama SQL*Plus. Básicamente, es un intérprete SQL con algunas opciones de edición y formateo de resultados.

Para poder acceder a una base de datos gestionada por ORACLE debemos ser un usuario autorizado de la misma y conocer la palabra clave, password, asociada al usuario.

Una vez iniciado SQL*Plus se pedirá el nombre de usuario, la password y por último la cadena de conexión.

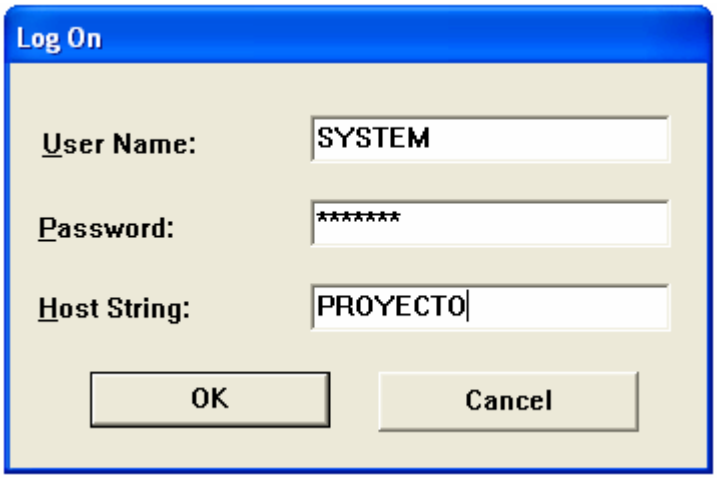

*Figura 16: Conexión SQL*Plus*

Una vez introducido los datos de correctos se abrirá una consola, desde la que se podrá manejar la base de datos.

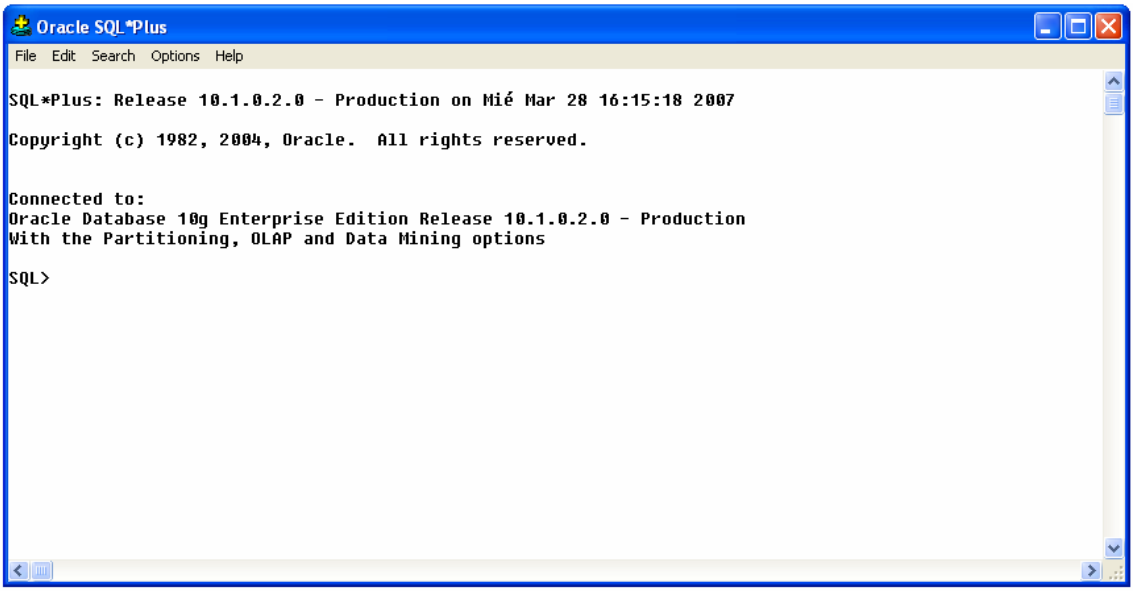

*Figura 17: Consola de SQL*Plus*

4.9 PL/SQL

PL/SQL es un lenguaje de programación estructurado. Es un lenguaje procedimental que amplía la funcionalidad de SQL, añadiendo estructuras habituales en otros lenguajes de programación, entre las que se encuentran:

- Variables y Tipos
- Estructuras de control
- Procedimientos y Funciones
- Tipos de Objetos y Métodos.

La unidad básica en PL/SQL es el bloque. Todos los programas PL/SQL están compuestos por bloques que pueden estar anidados. Un bloque PL/SQL está compuesto de tres partes principales:

- Sección declarativa (opcional). Contiene las variables, constantes, etc.
- Sección ejecutable (obligatoria). Contiene órdenes SQL para manipular datos de la base de datos y órdenes PL/SQL para manipular los datos del bloque.
- Sección de excepciones (opcional). Especifica las acciones a realizar en caso de error o cuando se producen excepciones en la ejecución.

La estructura general es:

[**DECLARE**

variables, constantes, excepciones de usuario...]

BEGIN

órdenes SQL

órdenes PL/SQL

[**EXCEPTION**

acciones a realizar al ocurrir un error]

END;

/

Para ejecutar un bloque PL/SQL siempre hay que colocar al final la barra /.

Podemos crear diferentes tipos de bloques:

- Bloques anónimos: Se construyen de forma dinámica y se suelen ejecutar una sola vez.
- Bloques nominados: Igual que los anónimos pero con una etiqueta que les da nombre.
- Subprogramas: Procedimientos, paquetes y funciones, almacenados en la base de datos y que se ejecutan en múltiples ocasiones. Los subprogramas se ejecutarán mediante una llamada.
- Disparadores ("Triggers"): Bloques nominados que se almacenan en la base de datos y que se ejecutan ante algún suceso.

En nuestro caso, PL/SQL ha sido enfocado en el uso de Disparadores.

5 MÉTODO DE RESOLUCIÓN

5.1 Requisitos Hardware y Software

El dominio de la aplicación se va a implementar en Oracle Database 10g versión 10.1.0.2.0 con la opción de Oracle Spatial instalada. Para poder llevar a cabo un buen funcionamiento de la aplicación son necesarias unas especificaciones en cuanto a hardware y software:

- ✓ Procesador Intel Pentium IV a 1,67 GHz o superior
- ✓ 512 Mb de RAM aunque es preferible 1Gb
- ✓ Sistema operativo Windows XP
- ✓ Visual Basic 6.0
- ✓ Microsoft Word XP

Para la realización de la interfaz se ha utilizado la aplicación Visual Basic 6.0. La aplicación Microsoft Word XP se ha utilizado para la redacción de la documentación del proyecto.

5.2 Dominio de la aplicación

En este apartado se realizará la especificación de los requisitos que se aplican al dominio de aplicación elegido.
En cuanto al modelo del "Control de vehículos" es necesario guardar la información de los vehículos que va a realizar los distintos recorridos que existen. De los vehículos interesa guardar la matrícula, el color y el modelo del mismo. La información más importante entre un vehículo y el recorrido que realiza son las horas en las que comienza y finaliza el recorrido. Estos tiempos se utilizan para calcular si el recorrido se ha realizado dentro del tiempo estimado según los tramos que componen el mismo. La granularidad de los tiempos utilizados es de horas, minutos y segundos. De un recorrido interesa guardar el identificador del mismo.

Respecto a los tramos, se ha distinguido entre los diferentes tramos de los que se compone cada recorrido. Existen varios tipos de tramos según la vía:

- Poblado
- Curva
- Recta

La diferencia entre los tres tipos de tramo radica en las velocidades mínimas y máximas permitidas. De esta forma dependiendo del tramo por el que se circule se deberán respetar unos límites u otros. Cada tipo de tramo lleva asociada una geometría que lo representa, así un poblado será un punto y la curva y recta representan lo que su nombre indica. De los tramos se desea guardar el identificador del tramo, la longitud, el tipo de tramo, su geometría, la velocidad aproximada a la que se circula, la velocidad mínima y la velocidad máxima.

Dentro de cada tramo existen puntos de control en los que se mide la velocidad que lleva un vehiculo en particular. Puede haber tantos puntos de control como se consideren necesarios, de esta forma dependiendo del tipo de tramo, se controlarán las velocidades de los vehículos en el momento en que pasen por dicho punto. Es importante saber el momento y la velocidad a la que pasa un determinado vehículo por un punto de control en concreto. De los puntos de control interesa almacenar el identificador de cada punto.

5.3 Diseño de la base de datos

A continuación se presenta el modelo de Entidad/Relación del sistema de "Control de Vehículos" y se explican cada una de las entidades, así como las relaciones existentes entre ellas.

5.3.1 Modelo Entidad / Relación

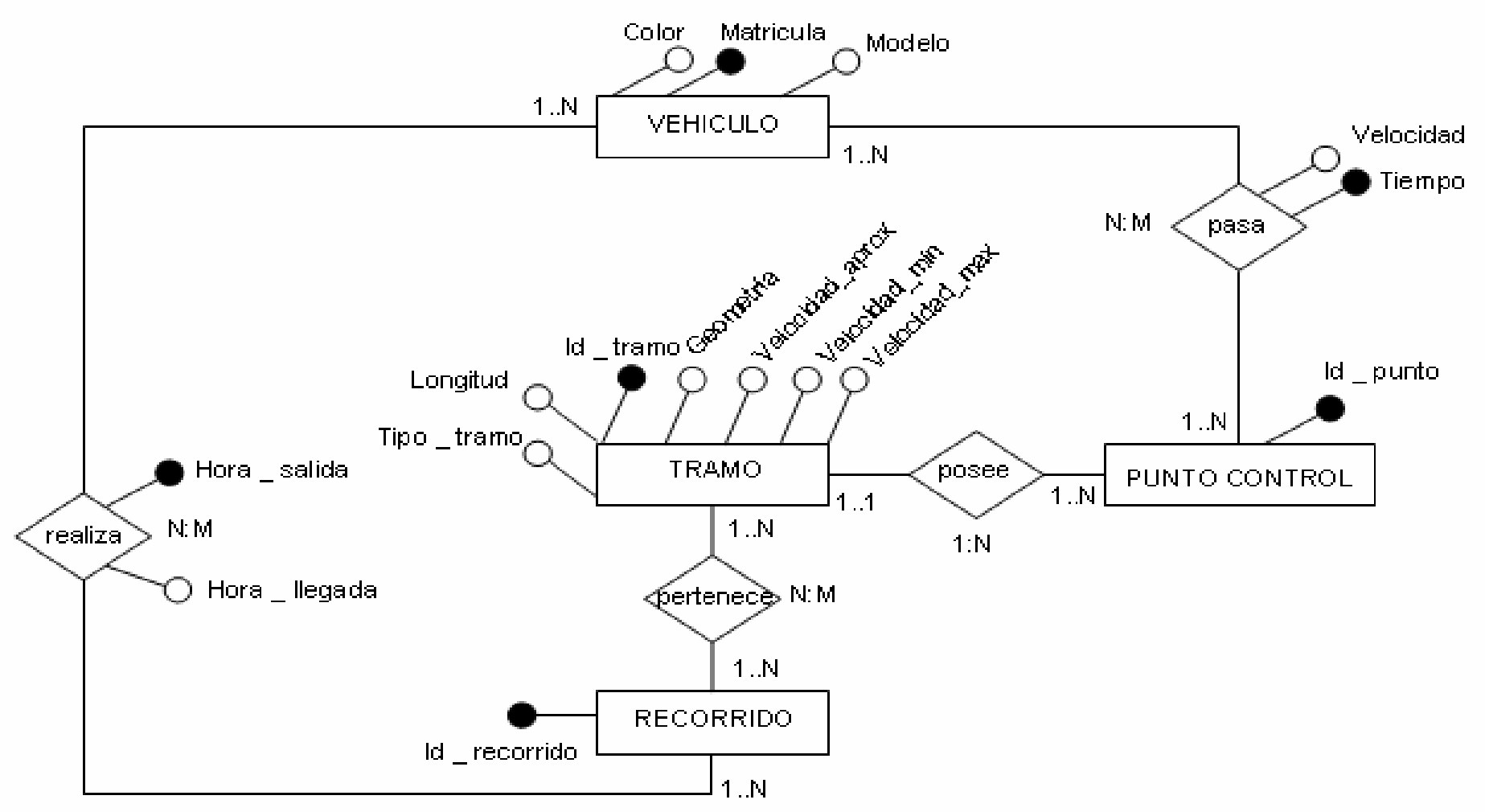

Ilustración 18: Modelo de Entidad/Relación

El primer paso es explicar la entidad Recorrido junto a su relación existente con la entidad Tramo. Como se puede observar la entidad recorrido posee un único campo obligatorio que es Id_recorrido, el cual identifica de forma única a un recorrido. Por otro lado la entidad Tramo, posee el campo Id_tramo, que a su vez identifica de forma única a un tramo. Mediante estos dos campos se relacionan estas entidades. Además, la entidad Tramo posee otro campo que es tipo_tramo, que puede tener una serie de valores ya definidos: curva, recta o poblado. Estos tres tipos de tramo serán los únicos que se contemplarán en esta aplicación. El campo longitud, que representa la longitud en kilómetros del tramo, es otro de los campos que componen esta entidad. El siguiente campo es la geometría. Este campo depende del valor que posea el campo tipo_tramo. Es decir, en el caso en que el tipo de tramo sea poblado, la geometría que se utilizará será el punto. En el caso en que el tramo sea una recta o una curva la geometría que se utilizará será la línea. Los últimos tres campos que componen la entidad Tramo son la velocidad máxima, la velocidad mínima y la velocidad aproximada. Estos campos se rellenan teniendo en cuenta las distintas velocidades de un tramo. La velocidad máxima y mínima son las limitaciones del tramo. Y la velocidad aproximada es la velocidad a la que se cree que aproximadamente circularan los vehículos por dicho tramo. La relación existente entre estas dos entidades es que un tramo puede pertenecer a uno o varios recorridos, por lo tanto la cardinalidd de esta relación es de N:M, siendo la cardinalidad mínima 1.

A continuación se pasará a explicar la entidad Punto de control, la entidad Vehículo ya la relación entre ambas. Esta entidad posee un único campo llamado id_punto, que identifica unívocamente a un punto de control. La entidad Vehículo posee matrícula, que identifica de forma única a un vehículo. El campo color y el campo modelo son los otros dos campos que componen esta entidad. La relación entre estas dos entidades radica en que un vehículo pasa por uno o varios puntos de control, por lo tanto la cardinalidad de esta relación es de N:M, siendo la cardinalidad mínima 1. Este hecho queda registrado con los campos velocidad y tiempo. La velocidad que queda registrada es la velocidad a la que ha pasado el vehículo por un punto de control en concreto. El tiempo es el momento, almacenado como fecha y hora, en el que un vehículo ha pasado por un punto de control.

La relación entre la entidad Punto de control y la entidad Tramo consiste en que un tramo puede contener uno o varios puntos de control, pero a su vez, un punto de control sólo puede pertenecer a un solo tramo. Por lo tanto la cardinalidad de esta relación es de 1:N.

Por último la relación que existe entre la entidad Vehículo y la entidad Recorrido es que un vehículo realiza uno o varios recorridos. La cardinalidad de esta relación es de N:M, siendo la cardinalidad mínima 1. Este vehículo parte con una hora de salida y otra de llegada. Esto servirá para controlar el tiempo en que un vehículo realiza un recorrido.

5.3.2 Modelo Relacional

Para realizar la transformación al relacional han sido seguidas las reglas básicas de transformación al modelo lógico, por lo que únicamente serán comentados aquellos puntos que se consideren conflictivos.

A continuación, se muestra el modelo relacional completo que se ha obtenido.

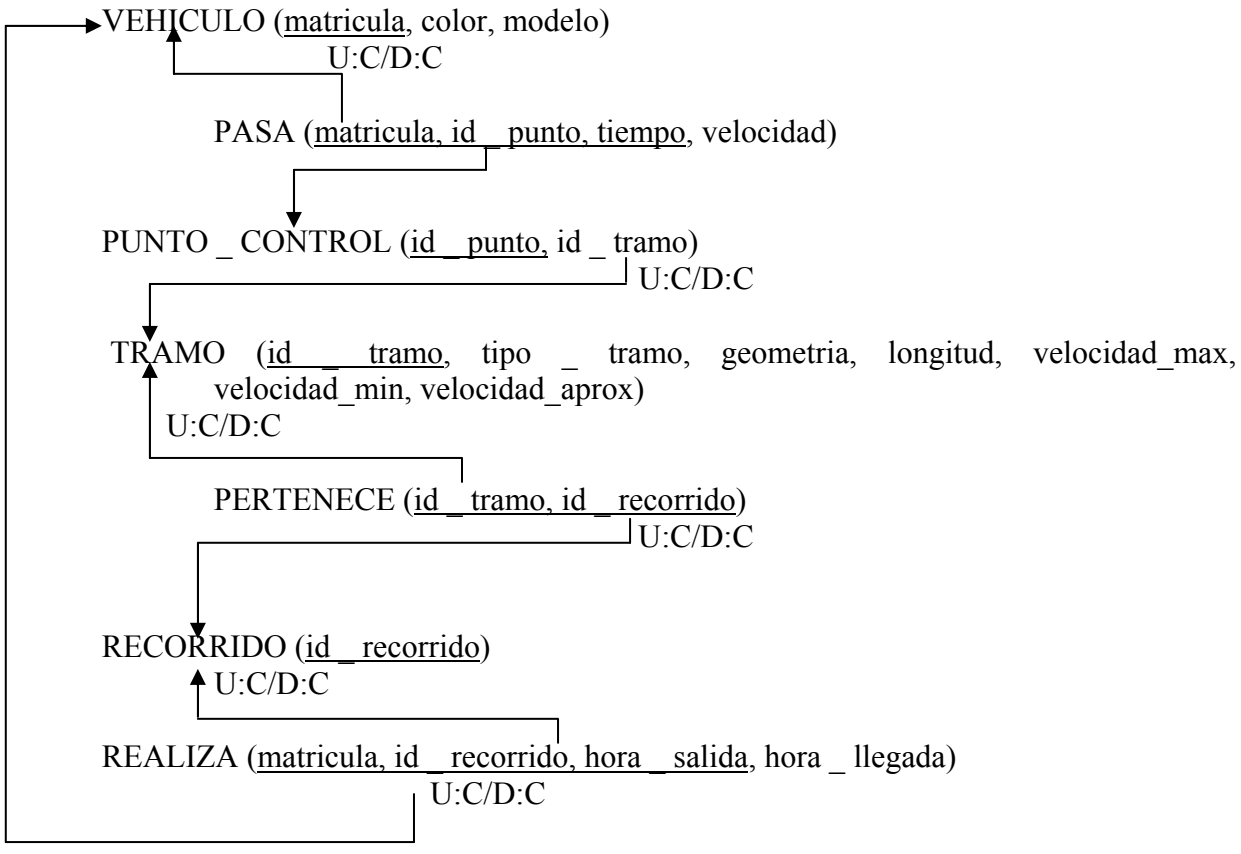

Ilustración 19 : Modelo Relacional

Como se observa en la figura 19, se ha producido algunos cambios respecto al modelo Entidad/Relación como puede ser la relación Posee que en este modelo ha pasado a convertirse en una propagación de clave desde la entidad Tramo a la entidad Punto Control.

Otra relación a tener en cuenta sería la existente entre la entidad Vehículo y la entidad Recorrido, ya que se tendrán tantas ocurrencias diferentes como situaciones distintas haya recogido en diferentes espacios de tiempo. Por lo tanto, se creará una relación dependiente de estas entidades.

La relación entre Tramo y Recorrido se compone de los campos id_tramo e id_recorrido que relaciona de forma unívoca el hecho de que un recorrido posea uno o varios tramos.
La relación entre la entidad Vehículo y la entidad Punto Control se traduce en una relación que contendrá tantas ocurrencias distintas como momentos en los que pase un vehículo por un punto de control en concreto.

- **Detalle de entidades, atributos y relaciones**

Entidad	Atributos	Relacionada con
Vehículo	**Matrícula***	**Punto Control**
	Color	**Recorrido**
	Modelo	
Tramo	**Id _tramo***	**Recorrido**
	Tipo_tamo	**Punto Control**
	Longitud	
	Geometría	
	Velocidad_max	
	Velocidad_min	
	Velocidad_aprox	
Recorrido	**Id_recorrido***	**Tramo**
		Vehículo

		Tramo
Punto Control	**Id_punto ***	
		Vehículo
	Matrícula*	
		Punto Control
Vehículo	**Color**	
		Recorrido
	Modelo	

Tabla 8: Definición de geometría Polígono con agujero

*** = Atributo Identificador Principal**

- **Diccionario de datos**

El modelo lógico contempla el siguiente diccionario de datos:

Diccionario de Datos Modelo Lógico

Relación Vehículo

Nombre Campo	Tipo de dato	Nulos	PK
matricula	varchar(10)	NOT NULL	Y
color	varchar(18)	NOT NULL	N
modelo	varchar(18)	NOT NULL	N

Tabla 9: Relación Vehículo

Relación Tramo

Nombre Campo	Tipo de dato	Nulos	PK
id_tramo	varchar(18)	NOT NULL	Y
tipo_tramo	varchar(10)	NOT NULL	N
geometria	MDSYS.SDO_GEOMETRY	NOT NULL	N
longitud	real	NOT NULL	N
velocidad_max	integer	NOT NULL	N
velocidad_min	integer	NOT NULL	N
velocidad_aprox	integer	NOT NULL	N

Tabla 10 : Relación Tramo

Relación Punto_Control

Nombre Campo	Tipo de dato	Nulos	PK
id_punto	varchar(18)	NOT NULL	Y
id_tramo	varchar(18)	NOT NULL	N

Tabla 11 : Relación Punto_Control

Relación Recorrido

Nombre Campo	Tipo de dato	Nulos	PK
id_recorrido	varchar(18)	NOT NULL	Y

Tabla 12 : Relación Recorrido

Relación Realiza

Nombre Campo	Tipo de dato	Nulos	PK
matricula	varchar(10)	NOT NULL	Y
id_recorrido	varchar(18)	NOT NULL	Y
hora_salida	timestamp	NOT NULL	Y
hora_llegada	timestamp	NOT NULL	N

Tabla 13 : Relación Realiza

Relación Pertenece

Nombre Campo	Tipo de dato	Nulos	PK
id_tramo	varchar(18)	NOT NULL	Y
id_recorrido	varchar(18)	NOT NULL	Y

Tabla 14 : Relación Pertenece

Relación Pasa

Nombre Campo	Tipo de dato	Nulos	PK
matricula	varchar(10)	NOT NULL	Y
id_punto	varchar(18)	NOT NULL	Y
tiempo	timestamp	NOT NULL	Y
velocidad	integer	NOT NULL	N

Tabla 15 : Relación Pasa

5.4 Disparadores

A continuación se pasará a explicar los distintos disparadores creados para controlar la aplicación:

5.4.1 CheckExcesoVelocidad

Este disparador comprueba que un vehículo no puede exceder de la velocidad permitida en un tramo en concreto. Como se ha explicado anteriormente en la entidad TRAMO existen unos campos que marcan las velocidades para cada tramo insertado. En el momento en el que se inserta un vehículo indicando la velocidad a la que pasa en un tramo en concreto este disparador es el que controlará que circule a una velocidad correcta.

```
CREATE or REPLACE TRIGGER CheckExcesoVelocidad

BEFORE INSERT ON PASA
FOR EACH ROW
DECLARE
   velocidadmax TRAMO.VELOCIDAD_MAX%TYPE;
   mat PASA.MATRICULA%TYPE;
   punto PASA.ID_PUNTO%TYPE;
   tiempo PASA.TIEMPO%TYPE;

BEGIN

SELECT PASA.MATRICULA,PASA.ID_PUNTO,PASA.TIEMPO INTO
mat,punto,tiempo FROM PASA
```

```
WHERE (PASA.MATRICULA=:new.MATRICULA AND
PASA.ID_PUNTO=:new.ID_PUNTO AND PASA.TIEMPO=:new.TIEMPO);


EXCEPTION

WHEN NO_DATA_FOUND THEN

SELECT TRAMO.VELOCIDAD_MAX INTO velocidadmax FROM TRAMO
WHERE TRAMO.ID_TRAMO=(SELECT PUNTO_CONTROL.ID_TRAMO
FROM PUNTO_CONTROL WHERE
PUNTO_CONTROL.ID_PUNTO=:new.ID_PUNTO);

IF velocidadmax<:new.VELOCIDAD THEN
/*HA SOBREPASADO LA VELOCIDAD*/

  RAISE_APPLICATION_ERROR(-20101,'%El vehiculo ha sobrepasado la
velocidad%');

ELSE
 /*VA A UNA VELOCIDAD CORRECTA*/

  RAISE_APPLICATION_ERROR(-20102,'%El vehiculo circula a una velocidad
correcta%');

END IF;

WHEN OTHERS THEN

     RAISE_APPLICATION_ERROR (-20100,'%La clave primaria ya existe%');

END;
```

5.4.2 CheckDosVehiculos

Este disparador comprueba que dos vehículos no están a la vez realizando el mismo recorrido. Se utilizan una serie de consultas para obtener los datos y realizar las comprobaciones.

```
CREATE or REPLACE TRIGGER CheckDosvehiculos

BEFORE INSERT ON REALIZA
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
DECLARE
  matricula REALIZA.MATRICULA%TYPE;
  hora REALIZA.HORA_SALIDA%TYPE;
```

```
BEGIN
matricula:='0';
SELECT REALIZA.MATRICULA,REALIZA.HORA_SALIDA INTO matricula,hora
FROM REALIZA WHERE (REALIZA.ID_RECORRIDO=:new.id_recorrido
AND (REALIZA.HORA_SALIDA<:new.HORA_SALIDA OR
REALIZA.HORA_SALIDA=:new.HORA_SALIDA) AND
(:new.HORA_SALIDA<REALIZA.HORA_LLEGADA OR
:new.HORA_SALIDA=REALIZA.HORA_LLEGADA));

IF matricula<>'0' and matricula<>:new.MATRICULA THEN

    RAISE_APPLICATION_ERROR (-20103, '%Error6%');

ELSE

IF matricula=:new.MATRICULA AND hora=:new.HORA_SALIDA THEN

    RAISE_APPLICATION_ERROR (-20101, '%Error5%');

END IF;

END IF;

EXCEPTION

WHEN NO_DATA_FOUND THEN

 DBMS_OUTPUT.PUT_LINE ('');

END;
```

5.4.3 CheckMismoVehiculo1

Disparador que comprueba que un vehículo no puede tener asignado dos recorridos distintos en el mismo día y en el mismo intervalo de tiempo.

```
CREATE or REPLACE TRIGGER CheckMismoVehiculo1

BEFORE INSERT ON REALIZA
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
DECLARE
  recorrido REALIZA.ID_RECORRIDO%TYPE;
  hora REALIZA.HORA_SALIDA%TYPE;

BEGIN
recorrido:=' ';
```

```
SELECT DISTINCT REALIZA.ID_RECORRIDO,REALIZA.HORA_SALIDA INTO
recorrido,hora FROM REALIZA WHERE
(REALIZA.MATRICULA=:new.MATRICULA
AND (REALIZA.HORA_SALIDA<:new.HORA_SALIDA OR
REALIZA.HORA_SALIDA=:new.HORA_SALIDA) AND
(:new.HORA_SALIDA<REALIZA.HORA_LLEGADA OR
:new.HORA_SALIDA=REALIZA.HORA_LLEGADA));

IF recorrido<>' ' AND recorrido<>:new.ID_RECORRIDO THEN

    RAISE_APPLICATION_ERROR(-20104, '%Error7%');

ELSE

  IF recorrido=:new.ID_RECORRIDO AND hora=:new.HORA_SALIDA THEN
  /*Se esta repitiendo la clave primaria*/
      RAISE_APPLICATION_ERROR (-20101, '%Error5%');
  ELSE

      IF recorrido=:new.ID_RECORRIDO AND hora<>:new.HORA_SALIDA THEN

          RAISE_APPLICATION_ERROR (-20104, '%Error7%');

      END IF;

  END IF;

END IF;

EXCEPTION

WHEN NO_DATA_FOUND THEN

    DBMS_OUTPUT.PUT_LINE ('');

END;
```

5.4.4 CheckMismoVehiculo2

Disparador complementario con el anterior para que un vehículo no pueda tener asignado el mismo recorrido en el mismo día y el mismo intervalo de tiempo.

```
CREATE or REPLACE TRIGGER CheckMismoVehiculo2

BEFORE INSERT ON REALIZA
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
DECLARE
```

```
  recorrido REALIZA.ID_RECORRIDO%TYPE;
  hora REALIZA.HORA_SALIDA%TYPE;

BEGIN
recorrido:=' ';

SELECT DISTINCT REALIZA.ID_RECORRIDO,REALIZA.HORA_SALIDA INTO
recorrido,hora FROM REALIZA WHERE
(REALIZA.MATRICULA=:new.MATRICULA
AND (REALIZA.HORA_SALIDA>:new.HORA_SALIDA OR
REALIZA.HORA_SALIDA=:new.HORA_SALIDA)
AND (:new.HORA_LLEGADA<REALIZA.HORA_LLEGADA OR
:new.HORA_LLEGADA=REALIZA.HORA_LLEGADA)
AND (:new.HORA_LLEGADA>REALIZA.HORA_SALIDA));

If recorrido<>' ' AND recorrido<>:new.ID_RECORRIDO THEN

    RAISE_APPLICATION_ERROR (-20105, '%Error8%');

ELSE

  IF recorrido=:new.ID_RECORRIDO AND hora=:new.HORA_SALIDA THEN
  /*Se esta repitiendo la clave primaria*/

      RAISE_APPLICATION_ERROR (-20101, '%Error5%');

  ELSE

    IF recorrido=:new.ID_RECORRIDO AND hora<>:new.HORA_SALIDA THEN

        RAISE_APPLICATION_ERROR (-20105, '%Error8%');
    END IF;

  END IF;

END IF;

EXCEPTION

WHEN NO_DATA_FOUND THEN

 DBMS_OUTPUT.PUT_LINE ('');

END;
```

5.4.5 Puntos

Este disparador comprueba si para unos datos de tiempos concretos el tramo se ha realizado a la velocidad permitida o bien se ha sobrepasado el límite de velocidad. Se utiliza una tabla auxiliar Fechas para realizar las comprobaciones.

```
CREATE or REPLACE TRIGGER Puntos

BEFORE INSERT ON REALIZA
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
DECLARE
  tiempomax REAL;
  tiempomin REAL;
  tiemporeal REAL;

BEGIN

SELECT (SUM (TRAMO.LONGITUD/TRAMO.VELOCIDAD_MAX)*60) INTO
tiempomin FROM TRAMO where id_tramo in
(select id_tramo from pertenece where pertenece.id_recorrido
=:new.ID_RECORRIDO);


SELECT (SUM (TRAMO.LONGITUD/TRAMO.VELOCIDAD_MIN)*60) INTO
tiempomax FROM TRAMO where id_tramo in
(select id_tramo from pertenece where pertenece.id_recorrido
=:new.ID_RECORRIDO);

INSERT INTO FECHAS (FECHA1,FECHA2) VALUES
(:new.HORA_LLEGADA,:new.HORA_SALIDA);

SELECT (SELECT EXTRACT (HOUR FROM (SELECT FECHA1-FECHA2 FROM
FECHAS)) FROM FECHAS)*60+
(SELECT EXTRACT (MINUTE FROM (SELECT FECHA1-FECHA2 FROM
FECHAS)) FROM FECHAS)+
(SELECT EXTRACT (SECOND FROM (SELECT FECHA2 FROM FECHAS))
FROM FECHAS)/60 INTO tiemporeal FROM FECHAS;

IF tiemporeal<tiempomax THEN

    RAISE_APPLICATION_ERROR (-20106,'%Error2%');

ELSE

    IF tiemporeal>tiempomin THEN

       RAISE_APPLICATION_ERROR (-20107,'%Error3%');

   ELSE

        RAISE_APPLICATION_ERROR (-20108,'%Error4%');
```

```
   END IF;
END IF;

DELETE FROM FECHAS;

END;
```

5.5 Consultas

5.5.1 Mismo recorrido

Consultar los vehículos que realizan el mismo recorrido.

```
SELECT MATRICULA, DISTINCT ID_RECORRIDO FROM REALIZA WHERE
ID_RECORRIDO IN (SELECT ID_RECORRIDO FROM REALIZA GROUP BY
ID_RECORRIDO HAVING COUNT(ID_RECORRIDO)>1)
```

5.5.2 Hora salida

Consultar la hora de salida de un vehículo para un recorrido.

```
SELECT HORA_SALIDA FROM REALIZA WHERE
REALIZA.MATRICULA='matrícula' AND REALIZA.ID_RECORRIDO='recorrido';
```

5.5.3 Tiempo mínimo

Calcular el tiempo mínimo que un vehículo tarda en realizar un recorrido.

```
SELECT (SUM (TRAMO.LONGITUD/TRAMO.VELOCIDAD_MAX)*60) AS
TIEMPO_MIN_MINUTOS FROM TRAMO where id_tramo in
(select id_tramo from pertenece where pertenece.id_recorrido = (select id_recorrido
from realiza where
matricula ='matrícula' and realiza.hora_Salida = 'hora_salida'));
```

5.5.4 Tiempo máximo

Calcular el tiempo máximo que un vehículo tarda en realizar un recorrido.

```
SELECT (SUM (TRAMO.LONGITUD/TRAMO.VELOCIDAD_MIN)*60) AS
TIEMPO_MAX_MINUTOS FROM TRAMO where id_tramo in
(select id_tramo from pertenece where pertenece.id_recorrido = (select id_recorrido
from realiza where
matricula ='matrícula' and realiza.hora_Salida = 'hora_salida'));
```

5.5.5 Tiempo aproximado

Calcular el tiempo aproximado que un vehículo tarda en realizar un recorrido.

```
SELECT (SUM (TRAMO.LONGITUD/TRAMO.VELOCIDAD_APROX)*60) AS
TIEMPO_APROX_MINUTOS FROM TRAMO where id_tramo in
(select id_tramo from pertenece where pertenece.id_recorrido = (select id_recorrido
from realiza where
matricula ='matrícula' and realiza.hora_Salida = 'hora_salida'));
```

5.5.6 Tiempo real

Calcular el tiempo real que un vehículo tarda en realizar un recorrido

```
SELECT DISTINCT
(SELECT DISTINCT
(SELECT DISTINCT EXTRACT (HOUR FROM
(SELECT (HORA_LLEGADA - HORA_SALIDA) FROM REALIZA WHERE
REALIZA.ID_RECORRIDO='recorrido' AND REALIZA.MATRICULA='matrícula'
AND REALIZA.HORA_SALIDA='hora_salida')) FROM REALIZA)*60 +
(SELECT DISTINCT EXTRACT (MINUTE FROM
(SELECT (HORA_LLEGADA - HORA_SALIDA) From REALIZA WHERE
REALIZA.ID_RECORRIDO='recorrido' AND REALIZA.MATRICULA='matrícula'
AND REALIZA.HORA_SALIDA='hora_salida')) FROM REALIZA) +
(SELECT DISTINCT EXTRACT (SECOND FROM
(SELECT (HORA_LLEGADA - HORA_SALIDA) From REALIZA WHERE
REALIZA.ID_RECORRIDO='recorrido' AND REALIZA.MATRICULA='matrícula'
AND REALIZA.HORA_SALIDA='hora_salida')) FROM REALIZA)/60
FROM REALIZA)AS TIEMPO_EN_MINUTOS_ FROM REALIZA
```

5.5.7 Puntos control

Consultar los puntos de control en los que un vehículo supera la velocidad permitida.

```
CREATE OR REPLACE VIEW PUNTOS_CONTROL
AS (SELECT DISTINCT
PASA.ID_PUNTO,TRAMO.tipo_tramo,PASA.VELOCIDAD,TRAMO.VELOCIDAD
_MAX FROM REALIZA INNER JOIN (PASA
INNER JOIN (PUNTO_CONTROL INNER JOIN TRAMO ON
(PUNTO_CONTROL.ID_TRAMO=TRAMO.ID_TRAMO))ON
(PASA.ID_PUNTO=PUNTO_CONTROL.ID_PUNTO)) ON
(PASA.MATRICULA=REALIZA.MATRICULA AND
REALIZA.MATRICULA='matrícula'AND
REALIZA.ID_RECORRIDO='recorrido' AND PASA.TIEMPO>'hora de salida' AND
PASA.TIEMPO<REALIZA.HORA_LLEGADA)) ORDER BY
```

```
ID_PUNTO;

SELECT * FROM PUNTOS_CONTROL WHERE
VELOCIDAD>VELOCIDAD_MAX;
```

5.6 Interfaz en Visual Basic 6.0

Para facilitar el uso de la aplicación por parte de usuarios se ha creado una interfaz en Visual Basic. A continuación se explicará una parte de la aplicación muy genérica ya que la explicación del funcionamiento de la misma se hará en el apartado de Experimentación.

Al comenzar la aplicación básicamente se tendrán tres opciones principales:

1. Insertar datos
2. Borrar datos
3. Consultas

A continuación se explicarán las tres opciones por separado.

5.6.1 Insertar datos

En esta opción se permite insertar datos dentro de las tablas que componen la base de datos. Hay que aclarar que la opción de inserción sólo se ofrece a las tablas principales, es decir, en las tablas auxiliares explicadas en el punto 5.3.2 no se pueden insertar nuevos datos.

Para cada tabla se mostrarán los campos obligatorios que la componen. En el caso de que se produzca algún error como un intento de duplicar la clave primara, la aplicación mostrará mensajes de error y no insertará datos dependiendo del caso en el que se produzcan.
Del mismo modo que se controla la no inserción de datos erróneos también se controlan los posibles avisos que se reciban por parte de los buscadores. En muchos casos estos avisos no son más que advertencias que no impiden la inserción de datos en la tabla

correspondiente. Estos casos también han sido controlados por la aplicación, mostrando el mensaje oportuno e insertando el dato.

5.6.2 Borrar datos

Esta opción permite borrar datos de las tablas que componen la base de datos. Al igual que en el punto anterior (Insertar datos), en esta opción no se permite eliminar datos de las tablas auxiliares.

La forma de borrar datos es una tabla que se carga con los datos de la tabla seleccionada. Esta "carga de datos" permite la opción de filtrado mediante los distintos campos que posee la tabla seleccionada.

Una vez seleccionado el registro a borrar, si se decide borrar el mismo, éste se borrará de forma permanente de la base de datos.

5.6.3 Consultas

Esta opción permite consultar los datos procedentes de las distintas consultas realizadas en la aplicación. En el caso de algunas consultas es necesario introducir datos para poder llevar a cabo la misma.

6 EXPERIMENTACIÓN

6.1 Manual de usuario

Para comenzar a utilizar la interfaz que se comunica con la base de datos, hay que ejecutar el archivo Proyecto localizado en la carpeta interfaz del proyecto.

Ilustración 20 : Icono del proyecto

Una vez ejecutado el archivo, aparecerá la pantalla de conexión con la base de datos.

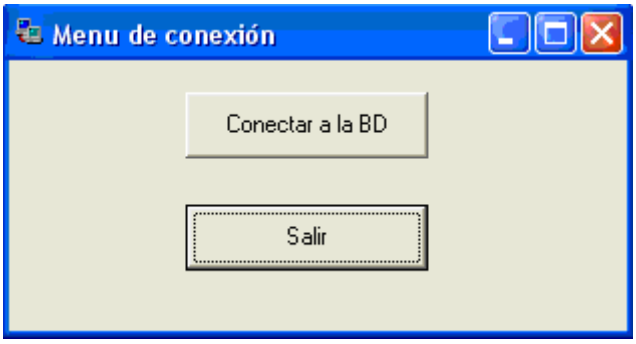

Ilustración 21: Conectar con la base de datos

Un punto importante es que desde cualquier parte de la aplicación se puede salir de la misma y desconectarse de la base de datos. O bien desde el botón Salir de esta pantalla o bien desde cualquiera de las pantallas de la aplicación pulsando la X de cerrar ventana. Cuando se pulsa una de estas dos opciones aparece una ventana esperando la confirmación para cerrar a conexión con la base de datos y cerrar la aplicación.

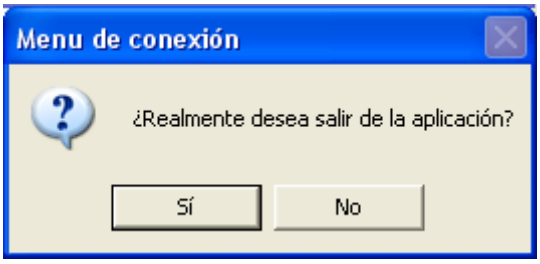

Ilustración 22: Salir de la aplicación

Si se continúa con la conexión y ésta se realiza con éxito, aparecerá una ventana informando de ello.

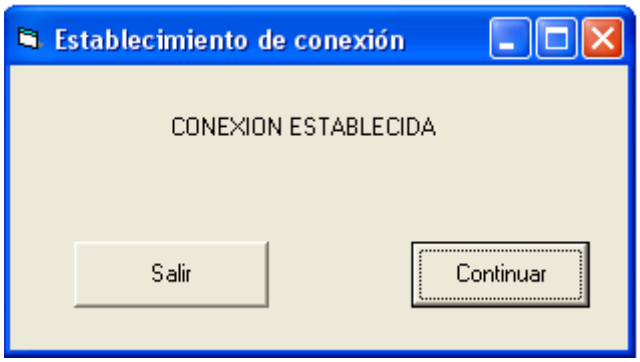

Ilustración 23: Conexión establecida

Una vez establecida la conexión, ya se puede comenzar a utilizar la aplicación como se ha explicado en los puntos de Inserción, Borrado y Consultas de datos.

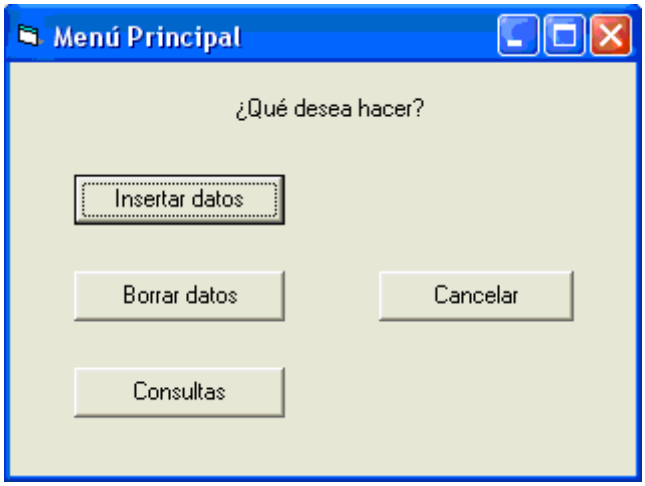

Ilustración 24: Menú principal

6.2 Pruebas de inserción de datos

A continuación se mostrarán las distintas interfaces para insertar datos en las diferentes tablas.

6.2.1 Insertar un vehículo

Lo primero que se debe hacer es seleccionar la tabla correspondiente a VEHICULO.

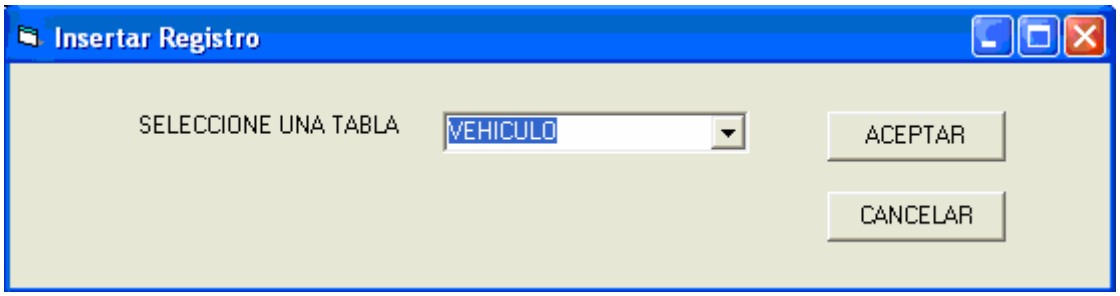

Ilustración 25 : Seleccionar tabla Vehículo

Una vez seleccionada la tabla correspondiente aparecerá un cuadro en el que se pedirán los datos correspondientes al vehículo. En el caso del campo matrícula existe una ayuda que permite conocer el formato necesario para dicho campo.

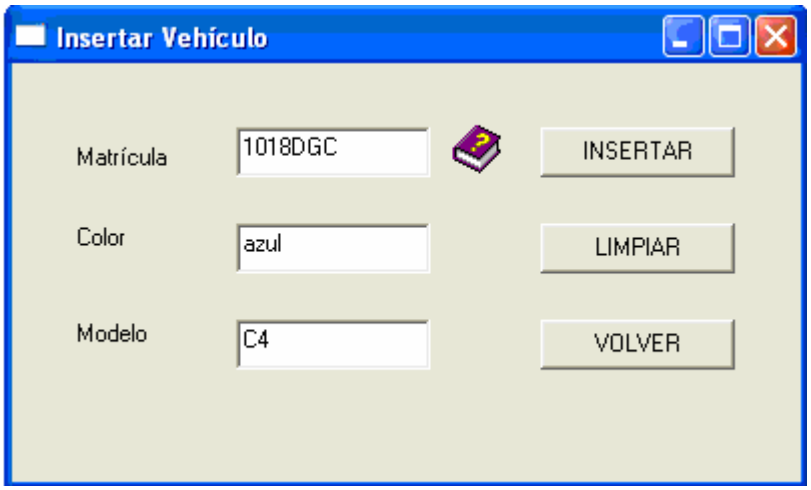

Ilustración 26 : Insertar vehículo

En este cuadro también se ofrecen las opciones de Limpiar y de Volver. La primera opción limpia todos los datos del cuadro. La opción volver regresa a cuadro de Insertar Registro.

En el caso en que no se rellene alguno de los campos, al ser todos ellos requeridos, se mostrará un mensaje de error indicando que se deben rellenar todos los campos.

Ilustración 27 : Error de campos obligatorios

La matrícula es la clave primaria de la tabla VEHICULO, por lo que si se intenta insertar dos registros con la misma matrícula la base de datos enviará un mensaje con el error. A continuación se le pedirá al usuario que introduzca una nueva matrícula.

Ilustración 28 : Error de clave única

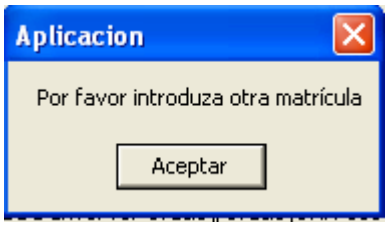

Ilustración 29 : Error matrícula

Una vez rellenados todos los campos correctamente el registro se insertará en la base de datos y se le mostrará un mensaje informando de ello al usuario.

Ilustración 30 : Vehículo insertado

6.2.2 Insertar un punto de control

Lo primero que se debe hacer es seleccionar la tabla correspondiente a PUNTO_CONTROL.

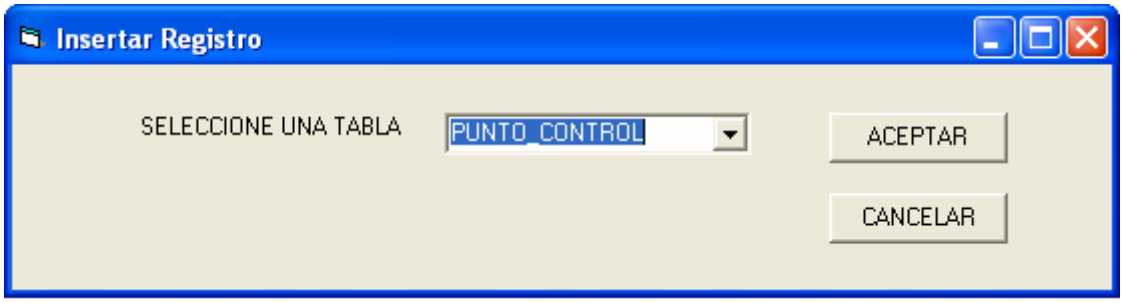

Ilustración 31 : Seleccionar tabla Punto de control

Una vez seleccionada la tabla correspondiente aparecerá un cuadro en el que se pedirán los datos correspondientes al punto de control.

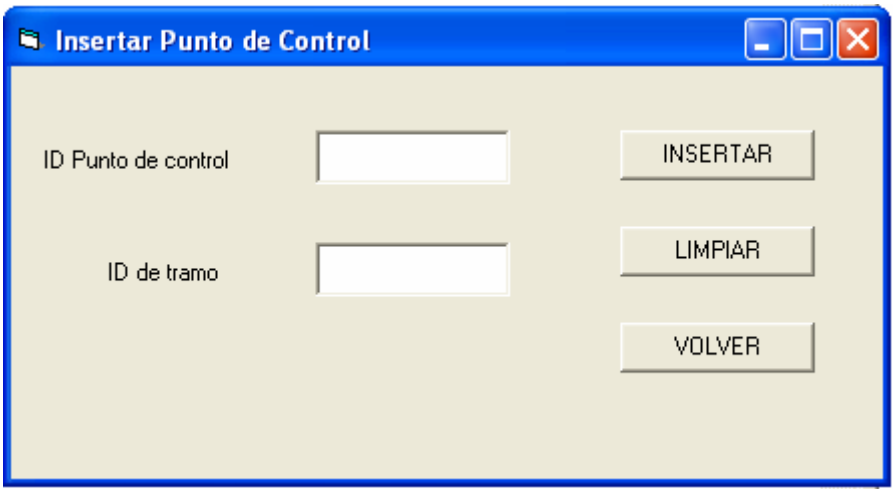

Ilustración 32 : Insertar punto de control

En este cuadro también se ofrecen las opciones de Limpiar y de Volver. La primera opción limpia todos los datos del cuadro. La opción volver regresa a cuadro de Insertar Registro.

En el caso en que no se rellene alguno de los campos, al ser todos ellos requeridos, se mostrará un mensaje de error indicando que se deben rellenar todos los campos.

Ilustración 33 : Error de campos obligatorios

El identificador del punto de control es la clave primaria de la tabla PUNTO_CONTROL, por lo que si se intenta insertar dos registros con el mismo identificador la base de datos enviará un mensaje con el error. A continuación se le pedirá al usuario que introduzca un nuevo punto.

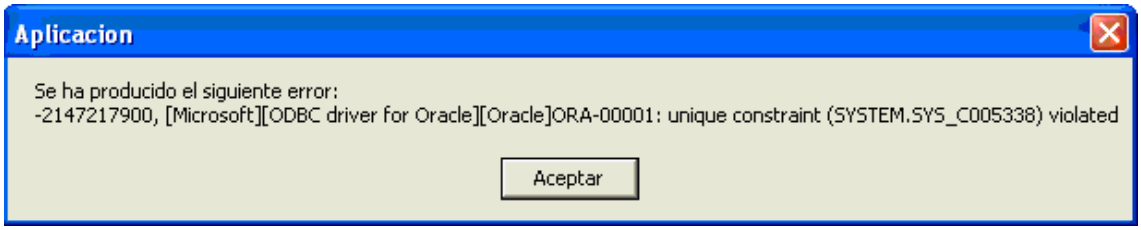

Ilustración 34 : Error de clave única

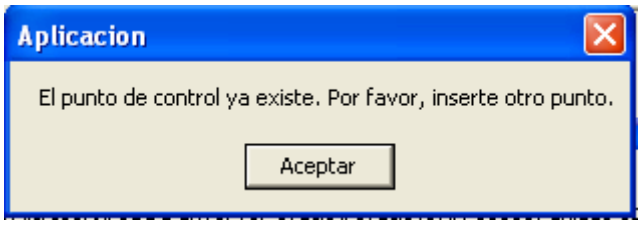

Ilustración 35 : Error insertar nuevo punto de control

Además de comprobar si el punto de control insertado es correcto, también hay que comprobar si el identificador del tramo introducido existe en la base de datos. En el caso en que no exista, se muestra un error y no se inserta el nuevo registro hasta no introducir todos los datos correctos.

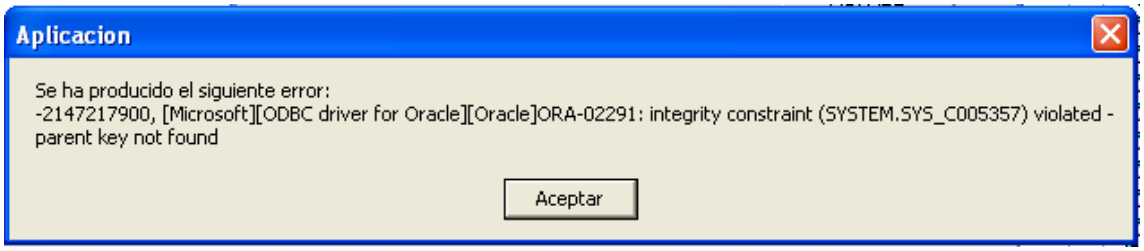

Ilustración 36 : Error de inserción

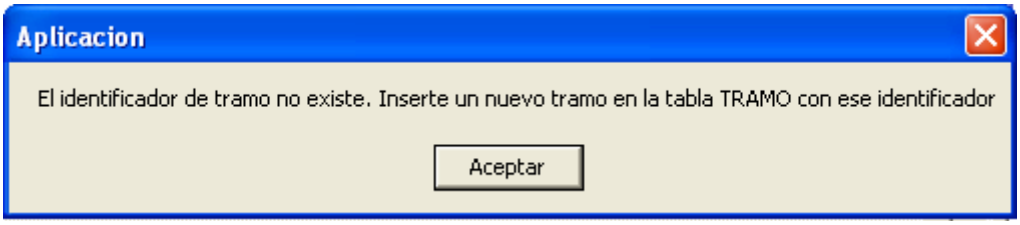

Ilustración 37 : Error de inserción

Una vez rellenados todos los campos correctamente el registro se insertará en la base de datos y se le mostrará un mensaje informando de ello al usuario.

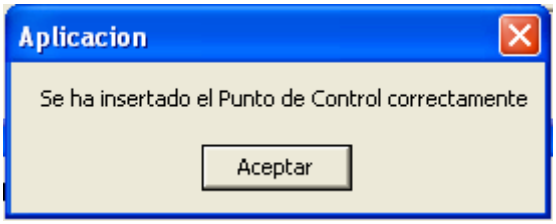

Ilustración 38 : Punto de control insertado

6.2.3 Insertar un tramo

Lo primero que se debe hacer es seleccionar la tabla correspondiente a TRAMO.

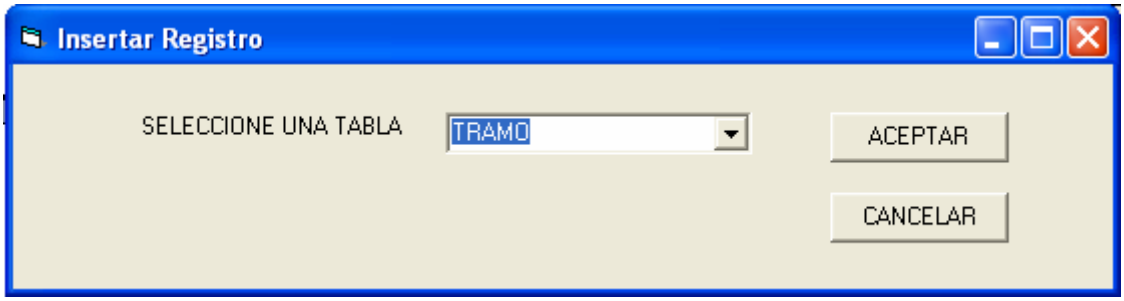

Ilustración 39 : Seleccionar tabla Tramo

Una vez seleccionada la tabla correspondiente aparecerá un cuadro en el que se pedirán los datos correspondientes al tramo.

Ilustración 40 : Insertar tramo

En el caso en que no se rellene alguno de los campos, al ser todos ellos requeridos, se mostrará un mensaje de error indicando que se deben rellenar todos los campos.

Ilustración 41 : Error de campos obligatorios

Como se ha explicado en capítulos anteriores, un tramo puede ser de tres tipos: curva, poblado o recta. La elección de un tipo u otro depende de la velocidad máxima y mínima de un tramo.

Ilustración 42 : Seleccionar tipo de tramo

Se controla que las velocidades sean números y no se permita introducir otro dato distinto.

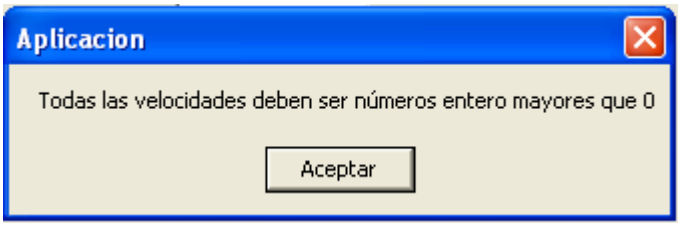

Ilustración 43 : Error al introducir velocidad

Igual que se controlan las velocidades se controla el tipo de dato correcto para la longitud.

Ilustración 44 : Error de longitud de entero

El identificador del tramo es la clave primaria de la tabla TRAMO, por lo que si se intenta insertar dos registros con el mismo identificador la base de datos enviará un mensaje con el error. A continuación se le pedirá al usuario que introduzca un nuevo tramo.

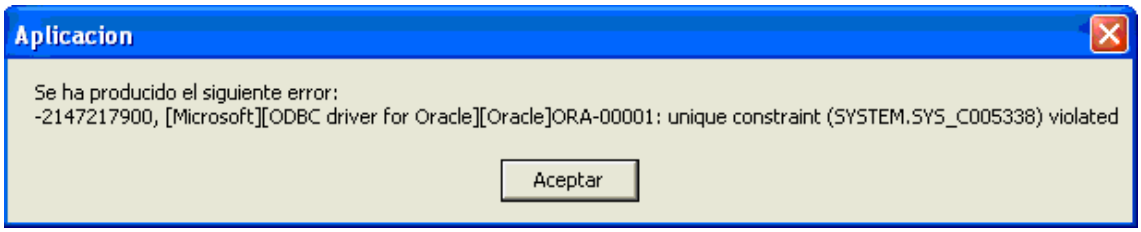

Ilustración 45 : Error de clave única

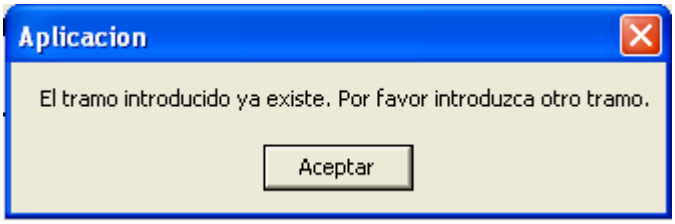

Ilustración 46 : Error de tramo ya existente

Una vez rellenados todos los campos correctamente el registro se insertará en la base de datos y se le mostrará un mensaje informando de ello al usuario.

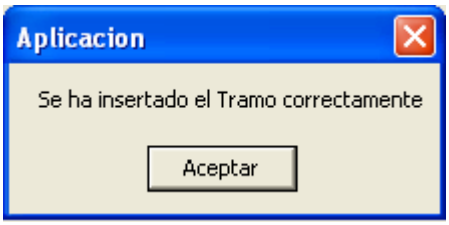

Ilustración 47 : Tramo insertado

6.2.4 *Insertar un recorrido*

Lo primero que se debe hacer es seleccionar la tabla correspondiente a RECORRIDO.

Ilustración 48 : Seleccionar tabla Recorrido

Una vez seleccionada la tabla correspondiente aparecerá un cuadro en el que se pedirán los datos correspondientes al recorrido.

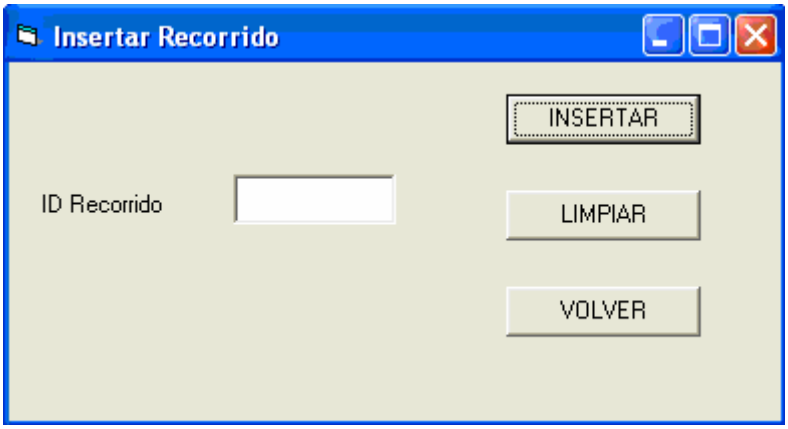

Ilustración 49 : Insertar recorrido

En este caso solo se pide que se rellene el identificador del recorrido, pero en el mismo caso que en los anteriores si no se rellena el campo se muestra un mensaje de error.

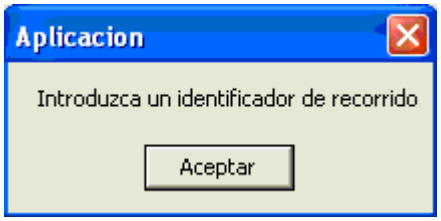

Ilustración 50 : Error identificador de recorrido

El identificador del tramo es la clave primaria de la tabla TRAMO, por lo que si se intenta insertar dos registros con el mismo identificador la base de datos enviará un mensaje con el error. A continuación se le pedirá al usuario que introduzca un nuevo tramo.

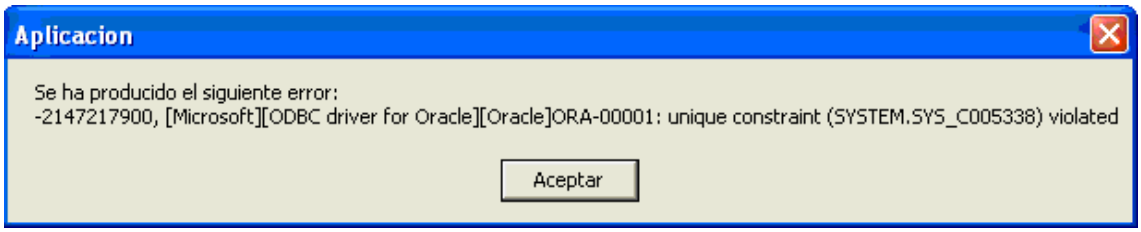

Ilustración 51 : Error de clave única

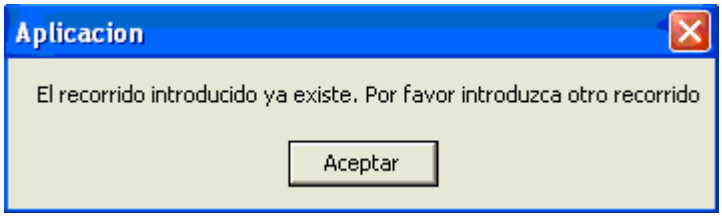

Ilustración 52 : Error recorrido ya existente

Una vez rellenados todos los campos correctamente el registro se insertará en la base de datos y se le mostrará un mensaje informando de ello al usuario.

Ilustración 53 : Recorrido insertado

6.2.5 Insertar cuando un vehículo pasa por un punto de control

Lo primero que se debe hacer es seleccionar la tabla correspondiente a PASA.

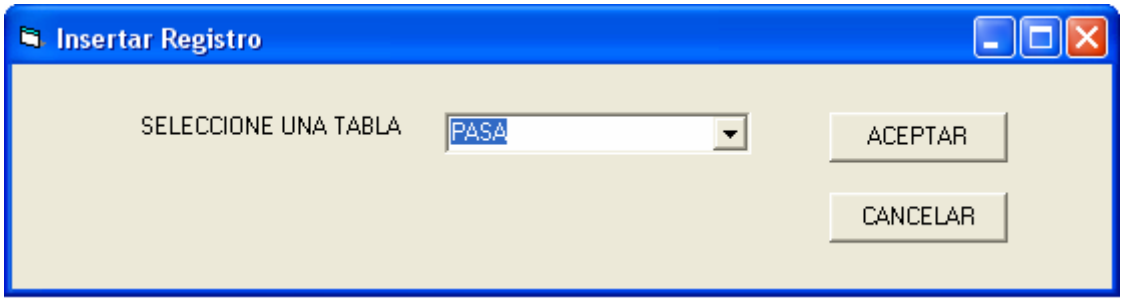

Ilustración 54 : Seleccionar tabla Pasa

Una vez seleccionada la tabla correspondiente aparecerá un cuadro en el que se pedirán los datos correspondientes para poder reflejar el paso de un vehículo por un punto de control.

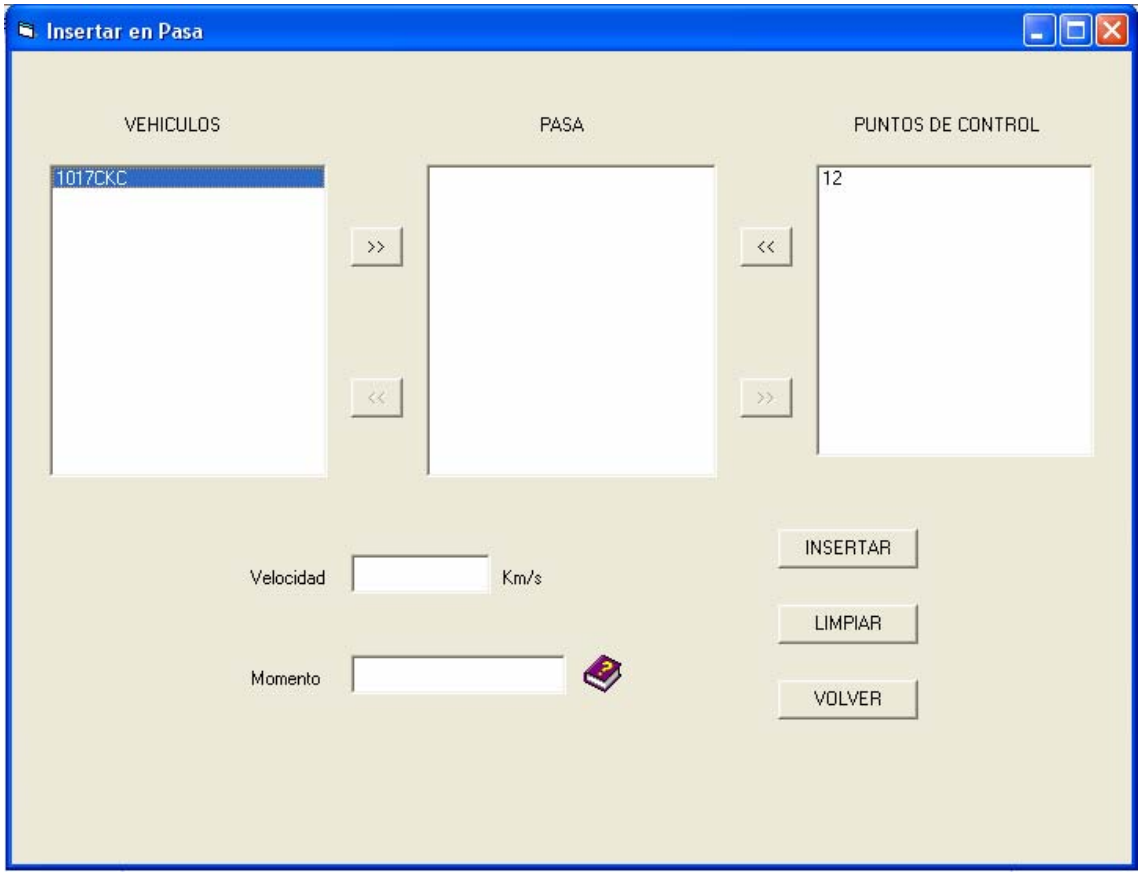

Ilustración 55 : Insertar en Pasa

Por un lado aparece un listado con todos los vehículos insertados en la base de datos. En este caso sólo se tiene un vehículo. En el otro lado aparecen los puntos de control existentes. Se podrá seleccionar únicamente en una misma inserción, un solo vehículo y un solo punto de control.

Del mismo modo que se seleccionan vehículos y puntos de control, también existe la opción de quitarlos de la selección.

Una vez seleccionados ambos datos, se deberá rellenar los campos de velocidad a la que pasa el vehículo por el punto de control y el momento en el que lo hace. Para saber el formato del momento a introducir, existe una ayuda disponible para el usuario, tal y como aparece en la siguiente imagen.

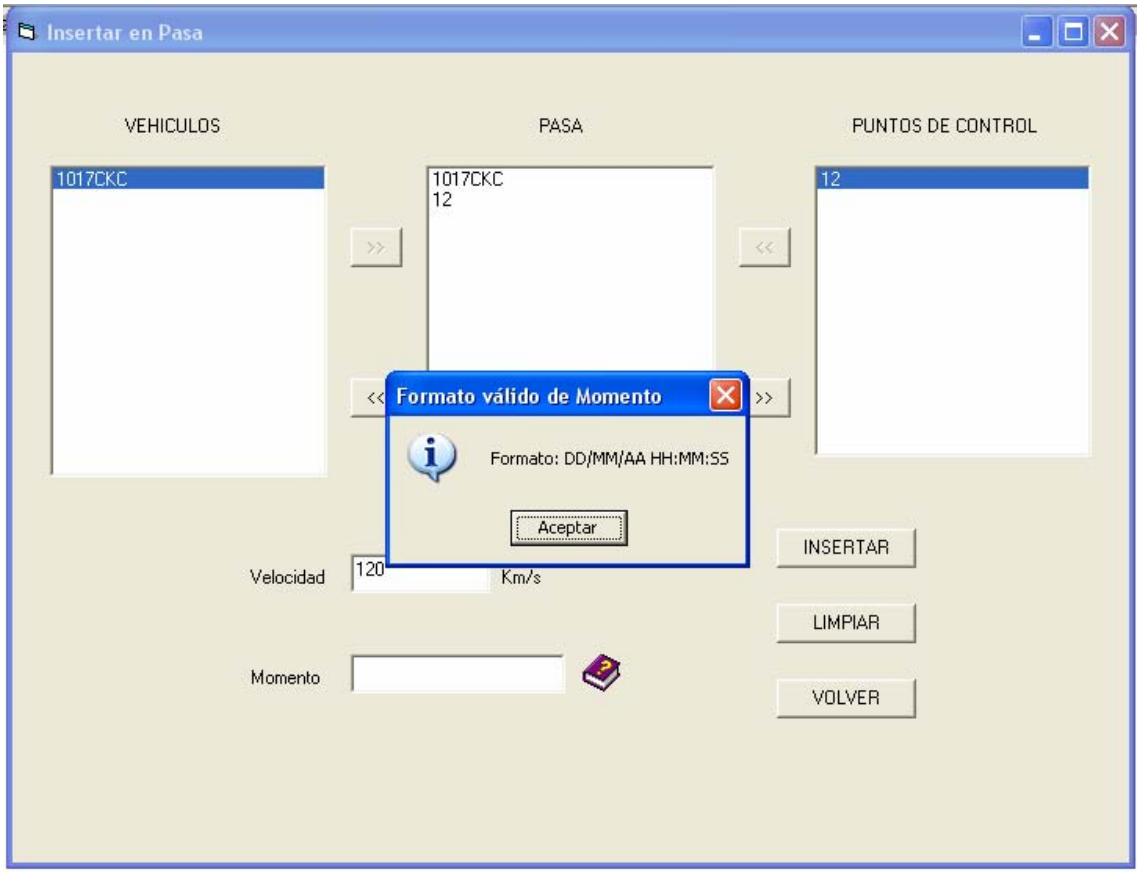

Ilustración 56 : Error de formato

La siguiente imagen muestra la forma en que se deben introducir los datos.

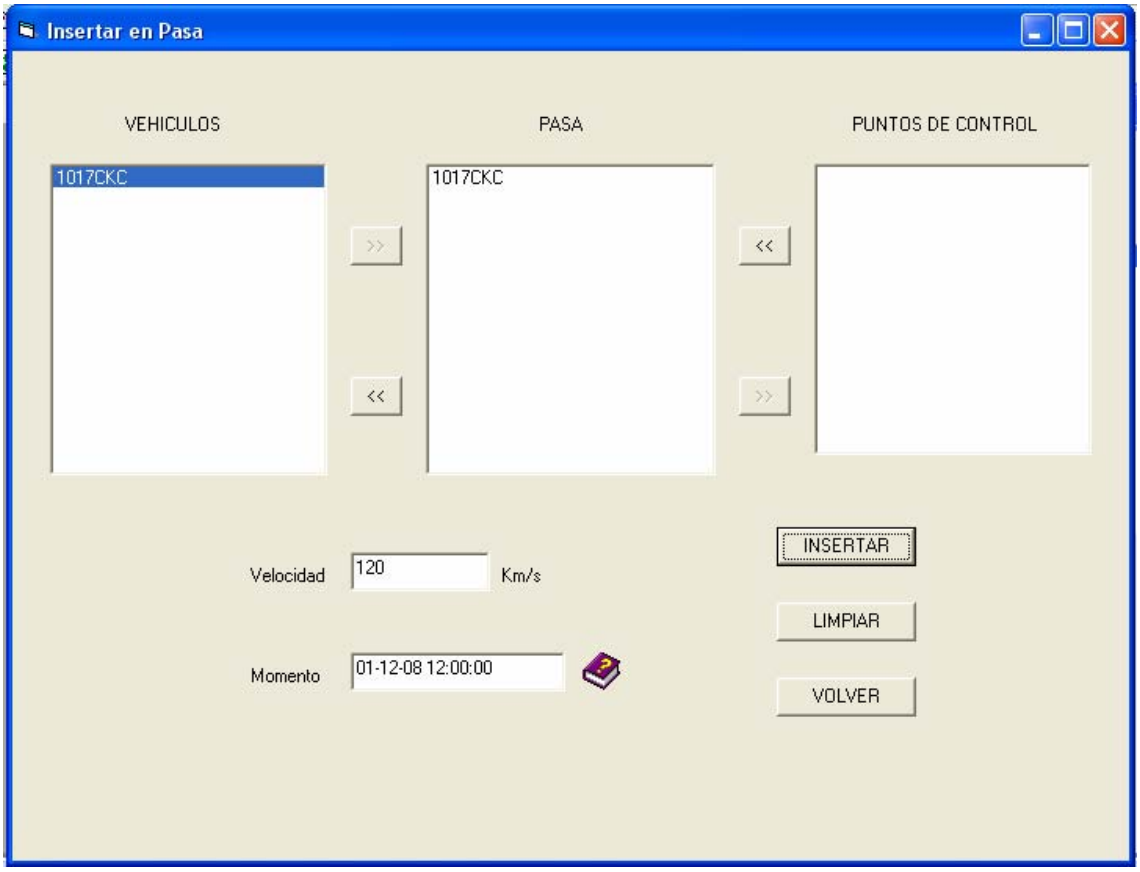

Ilustración 57 : Insertar datos en Pasa

Una vez insertados los datos aparece un cuadro de diálogo con la información proporcionada por un disparador acerca de la velocidad a la que pasa el vehículo por el punto de control. Si la velocidad a la que circula el vehículo es inferior o igual a la que está permitida circular en dicho punto aparecerá el siguiente cuadro.

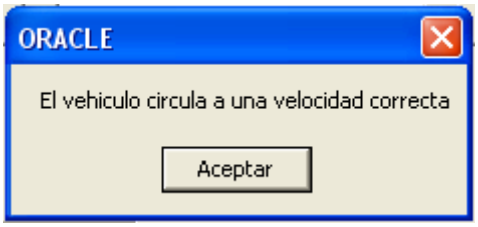

Ilustración 58 : Velocidad correcta

Posteriormente se informa de que la inserción del registro se ha realizado con éxito.

Ilustración 59 : Registro insertado en Pasa

Algo a tener en cuenta son los registros repetidos dentro de la tabla PASA. En el caso en que se intente insertar dos registros con la misma clave primaria, se produce un error como el que se muestra a continuación, indicando los datos que componen la clave primaria.

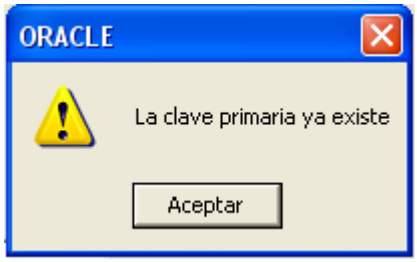

Ilustración 60 : Error de clave primaria

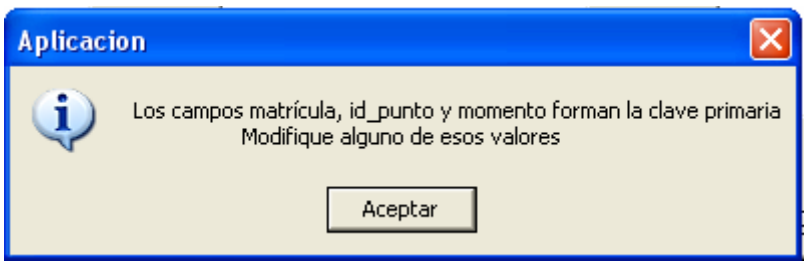

Ilustración 61 : Error de clave única

Si se introduce una velocidad mayor que la permitida en el punto de control seleccionado, un disparador devuelve un aviso con el hecho de que el vehículo supere la velocidad.

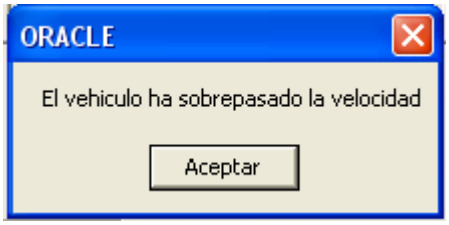

Ilustración 62 : Velocidad sobrepasada

Aunque el vehículo sobrepase la velocidad el registro se inserta en la base de datos.

6.2.6 Insertar un punto de control perteneciente a un tramo

Lo primero que se debe hacer es seleccionar la tabla correspondiente a PERTENECE.

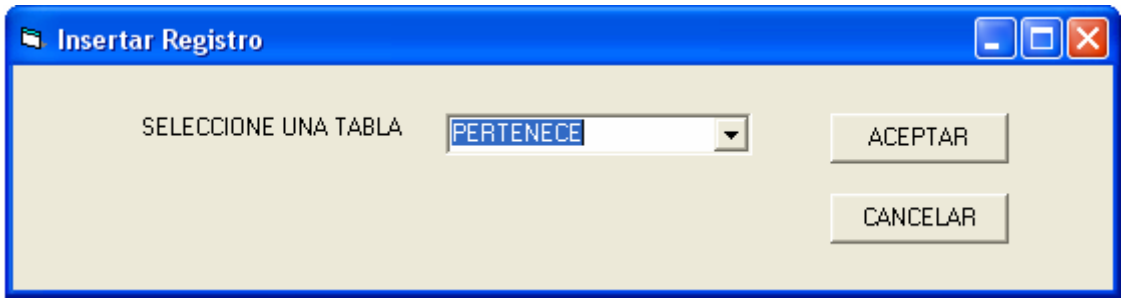

Ilustración 63 : Seleccionar tabla Pertenece

Una vez seleccionada la tabla correspondiente aparecerá un cuadro en el que se pedirán los datos correspondientes a la tabla pertenece.

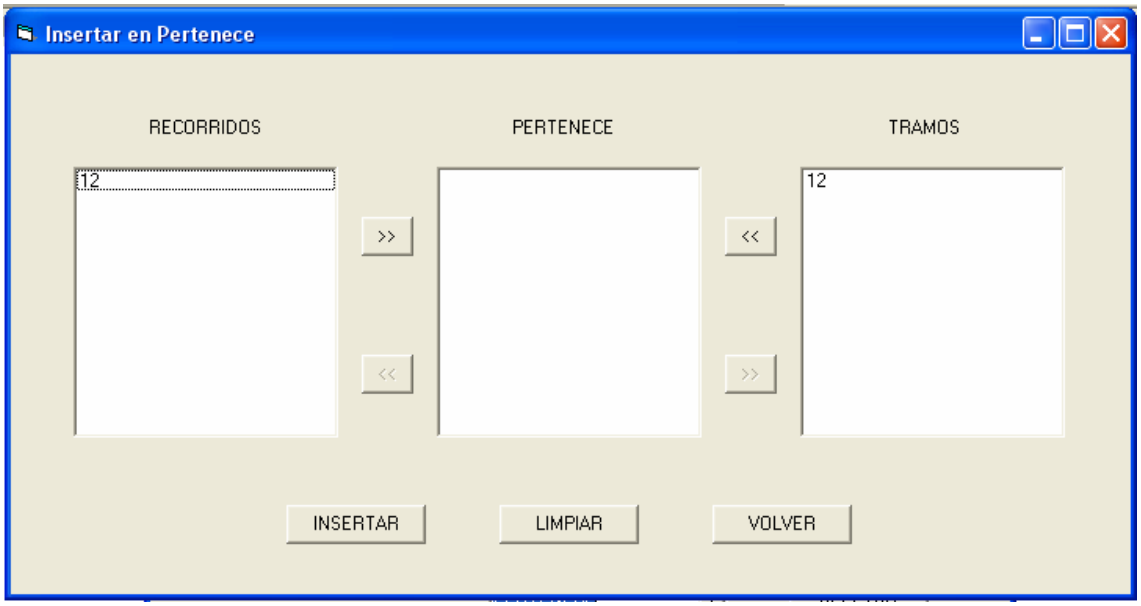

Ilustración 64 : Insertar tabla Pertenece

Por un lado aparecen los recorridos que existen en la base de datos y por otro lado los tramos que también se encuentran en la misma. Se debe seleccionar un solo recorrido y un solo tramo. En el caso en que se intente insertar una clave primaria repetida se mostrarán los siguientes errores.

Ilustración 65 : Error de clave única

Ilustración 66 : Error de clave primaria

Una vez seleccionados los datos correctos se insertará el registro en la base de datos y se mostrará un cuadro indicando este hecho.

Ilustración 67 : Registro insertado en Pertenece

6.2.7 Insertar el recorrido que realiza un vehículo

Lo primero que se debe hacer es seleccionar la tabla correspondiente a REALIZA.

Ilustración 68 : Seleccionar tabla Realiza

Una vez seleccionada la tabla correspondiente aparecerá un cuadro en el que se pedirán los datos correspondientes a la tabla realiza.

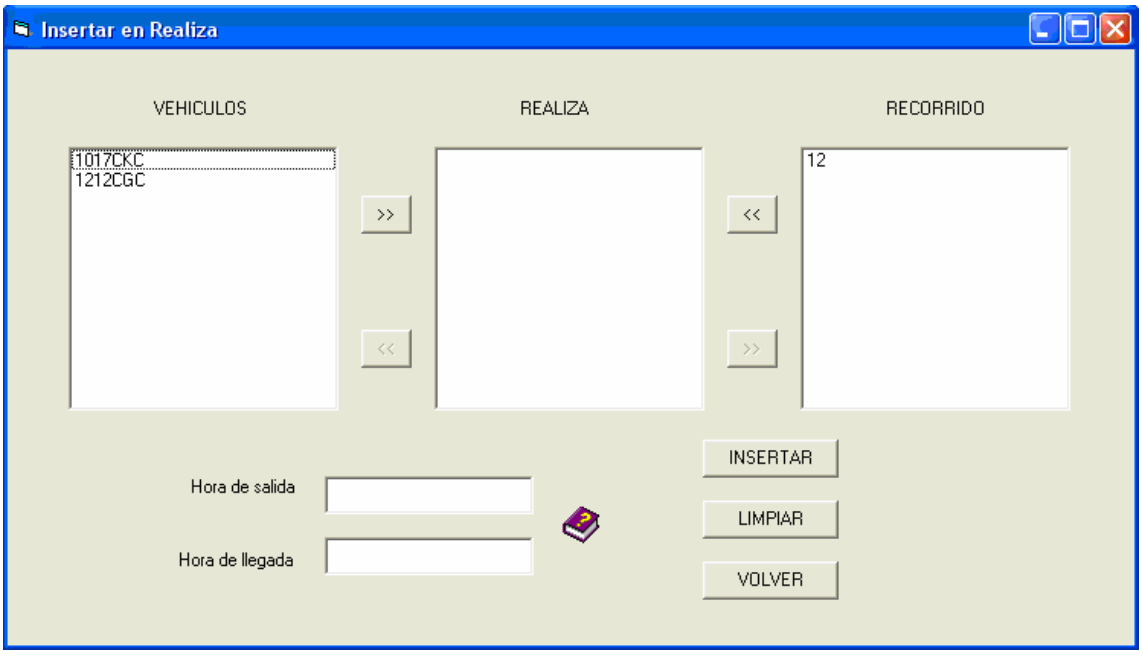

Ilustración 69 : Insertar datos en Realiza

Por un lado se tiene un listado con todos los vehículos que se encuentran en la base de datos, y por otro lado los recorridos que existen en la misma. Se debe seleccionar sólo un vehículo por recorrido que realiza. Una vez seleccionado el vehículo y el recorrido, se deberán insertar los datos correspondientes a la hora de salida del vehículo y la hora de llegada. Dependiendo de los tiempos introducidos, un disparador indicará la situación del vehículo respecto al recorrido realizado.

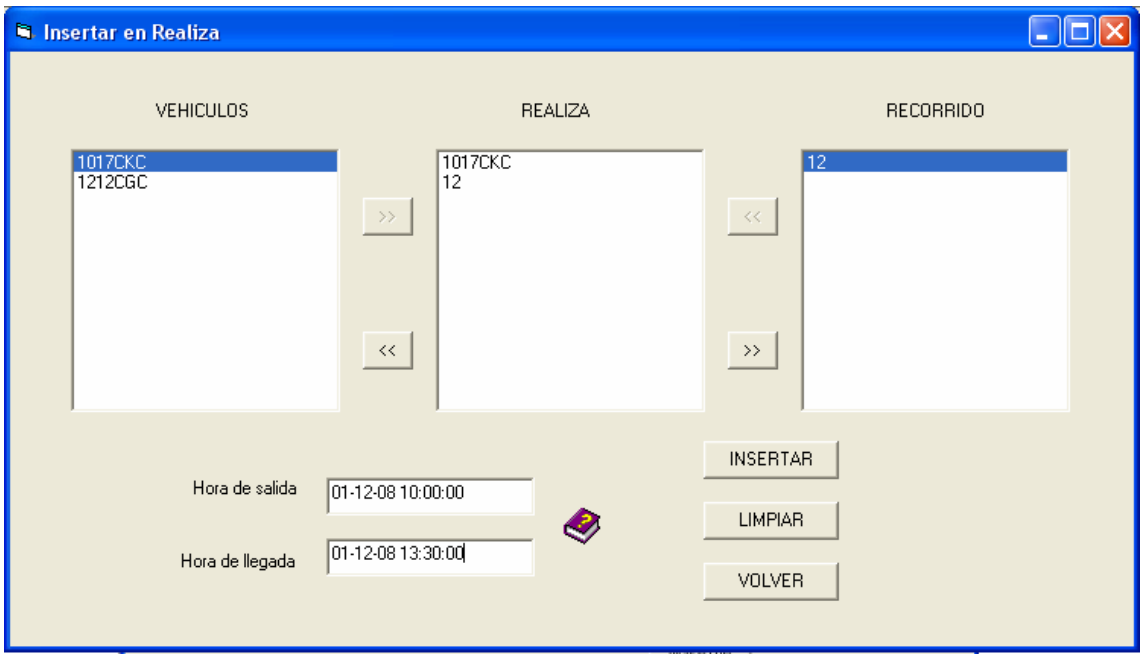

Ilustración 70 : Insertar registro en Realiza

Si se introducen de forma errónea las fechas, se muestra un mensaje indicándolo.

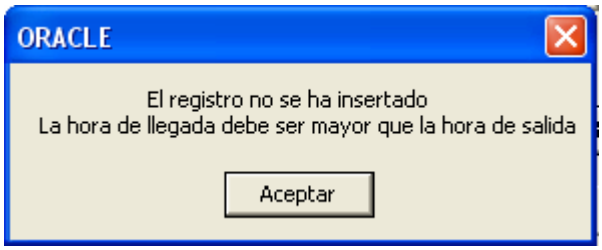

Ilustración 71 : Error de hora llega y de salida

Si se intenta introducir dos veces el mismo registro dentro de la tabla realiza, se muestra un mensaje de error indicándole al usuario que debe modificar sus datos de inserción.

Ilustración 72 : Error de clave primaria

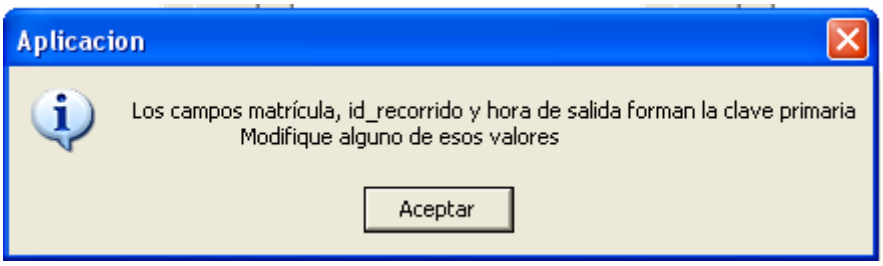

Ilustración 73 : Error de clave primaria

Si una vez introducidos los tiempos, el tiempo total (tiempo real) que ha tardado el vehículo en realizar el recorrido, según los datos insertados, es mayor que el tiempo calculado dependiendo de los tramos que conforman el recorrido, se muestra un mensaje indicando que el vehículo ha tardado más tiempo del estimado en realizar el recorrido. Aunque se produzca esta advertencia, el registro se inserta en la base de datos.

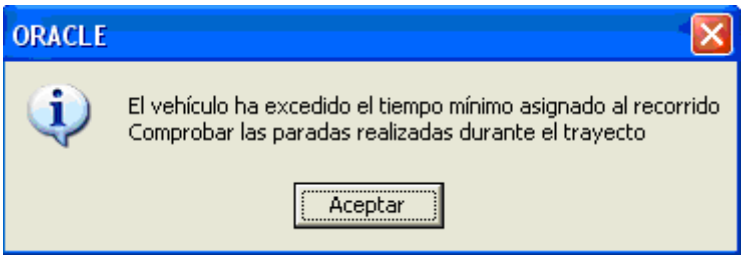

Ilustración 74 : Tiempo mínimo excedido

Si el recorrido ya está asignado a otro vehículo dentro de un mismo intervalo de tiempo, se muestra un mensaje de error indicando al usuario que el registro no se ha insertado en la base de datos debido al error.

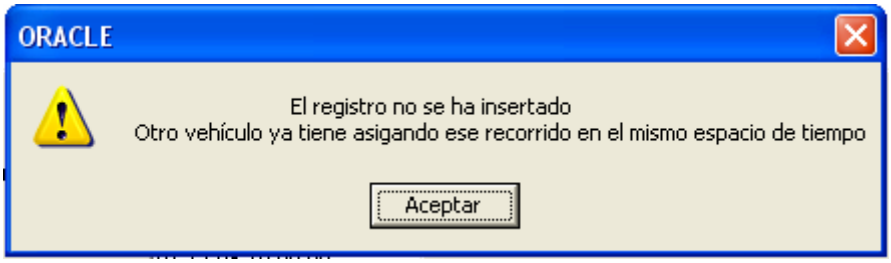

Ilustración 75 : Otro vehículo con recorrido ya asignado

Si el vehículo seleccionado ya tiene asignado otro recorrido en el mismo espacio de tiempo que el que se ha indicado, se muestra un mensaje de error. El registro no se inserta en la base de datos.

Ilustración 76 : Vehículo con recorrido ya asignado

Si los tiempos insertados entran dentro del intervalo correcto calculado, se muestra un mensaje indicándolo. A su vez el registro se inserta en la base de datos.

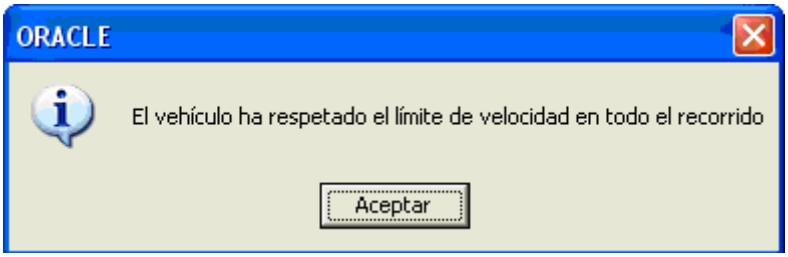

Ilustración 77 : Límite de velocidad respetado

Ilustración 78 : Registro insertado en Realiza

6.3 Pruebas de borrado de datos

Antes de explicar las pruebas de borrado de datos, se debe indicar que al ser borrados en cascada, el orden de borrado de las tablas se produce de forma coherente. Es decir, si existen datos de un vehículo que pasa por un punto de control, si se procede a borrar el registro de dicho vehículo, automáticamente se produciría un borrado en cascada que borraría todos los registros en los que estuviera presente dicho vehículo.

6.3.1 Borrar un vehículo

Lo primero que se debe hacer es seleccionar la tabla correspondiente a VEHICULO.

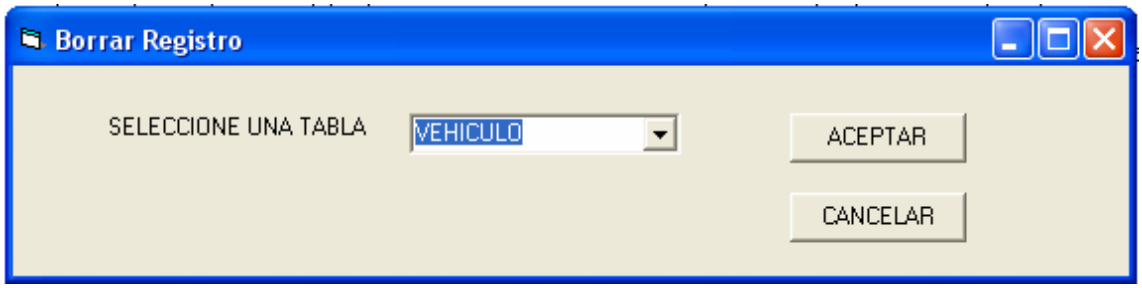

Ilustración 79 : Borrado. Seleccionar tabla Vehículo

Una vez seleccionada la tabla, se muestra la siguiente pantalla. Para obtener los datos de la tabla VEHÍCULO, se debe pulsar el botón de Cargar Datos, en el caso en que existan datos dentro de la tabla se cargaran dichos datos en la tabla que aparece. El botón de Limpiar Datos borra de la tabla los datos mostrados, pero no de la base de datos. El boton Volver regresa a la pantalla anterior. Si se pulsa el botón de Filtrar Búsqueda se extiende una parte del formulario que permite filtrar la búsqueda que se quiere realizar por los campos que componen la tabla VEHÍCULO.

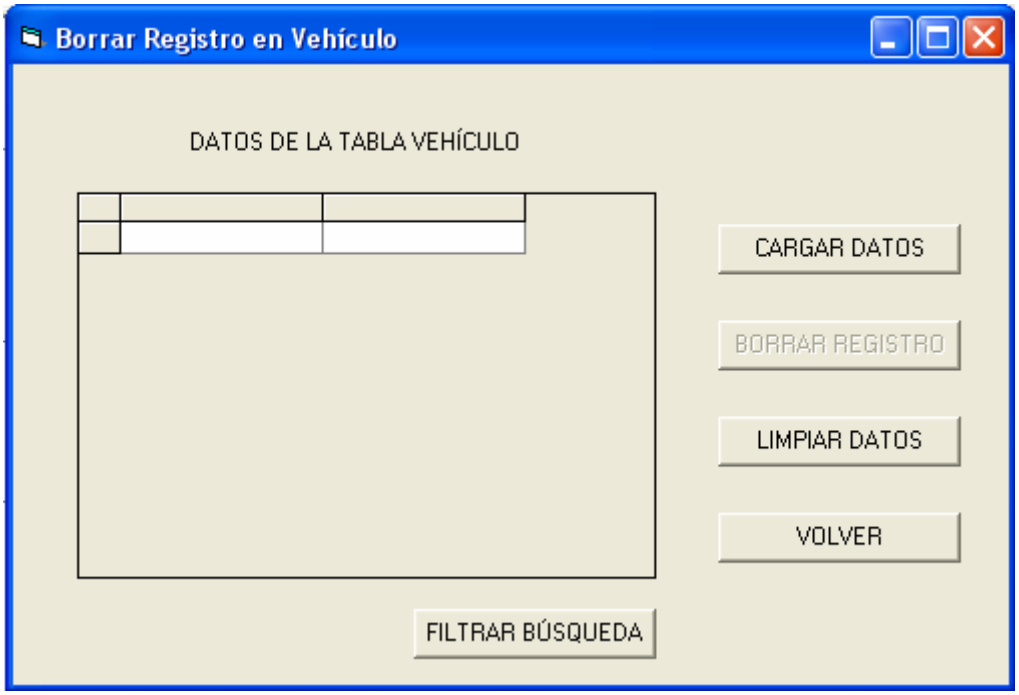

Ilustración 80 : Buscar en Vehículo

En el caso en que no se encuentren datos a mostrar en la tabla de VEHÍCULO se muestra el siguiente mensaje.

Ilustración 81 : Tabla vacía

Si se han encontrado datos, se muestran del modo que indica la siguiente pantalla.

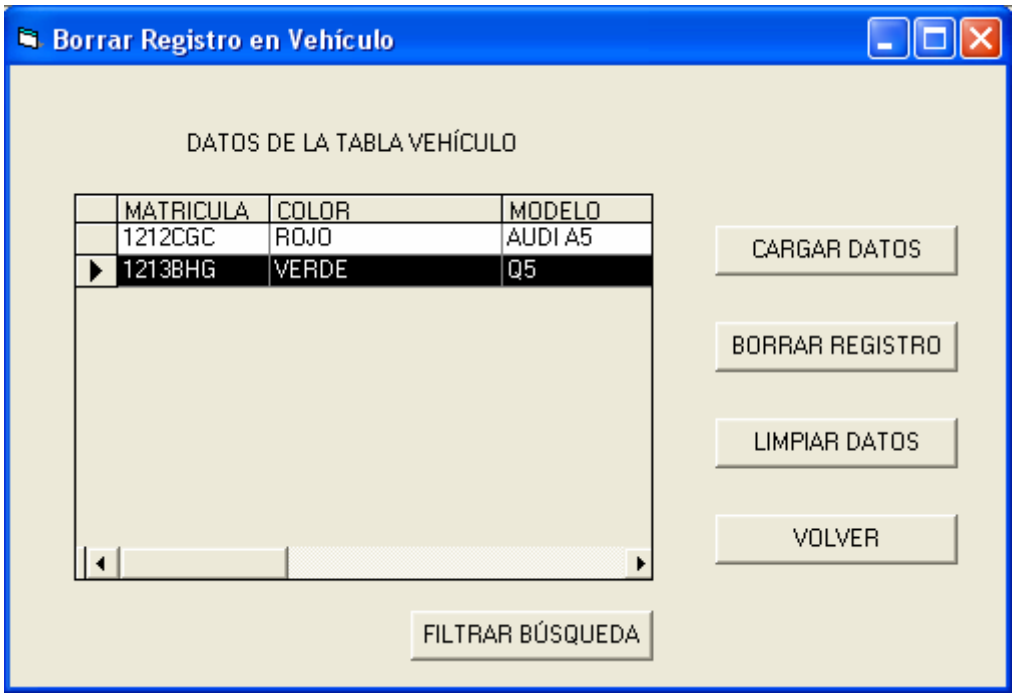

Ilustración 82 : Resultado de búsquedas en Vehículo

La opción de Filtrar Búsqueda muestra las siguientes posibilidades:

Ilustración 83 : Seleccionar vehículo

Una vez seleccionado el registro a eliminar se muestra un mensaje para asegurarse que se quiere eliminar el registro de la base de datos.

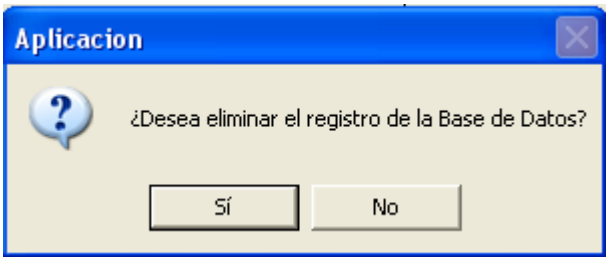

Ilustración 84 : Confirmar borrado

6.3.2 Borrar un punto de control

Lo primero que se debe hacer es seleccionar la tabla correspondiente a PUNTO DE COTROL.

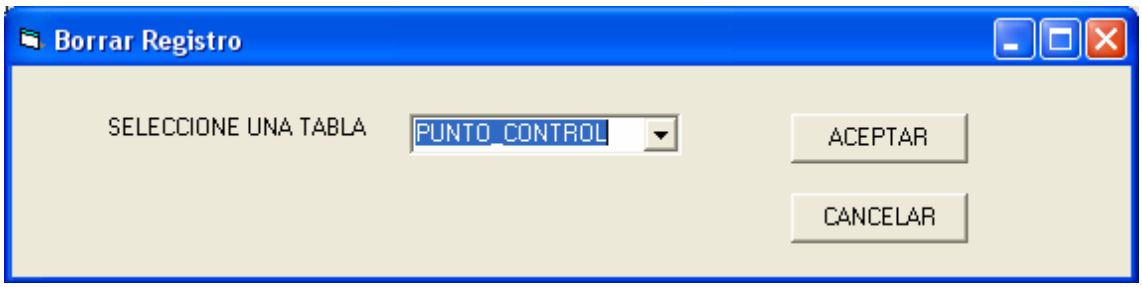

Ilustración 85 : Borrado. Seleccionar tabla Punto Control

Una vez seleccionada la tabla, se muestra la siguiente pantalla. Para obtener los datos de la tabla PUNTO DE CONTROL, se debe pulsar el botón de Cargar Datos, en el caso en que existan datos dentro de la tabla se cargaran dichos datos en la tabla que aparece. El botón de Limpiar Datos borra de la tabla los datos mostrados, pero no de la base de datos. El botón Volver regresa a la pantalla anterior. Si se pulsa el botón de Filtrar Búsqueda se extiende una parte del formulario que permite filtrar la búsqueda que se quiere realizar por los campos que componen la tabla PUNTO DE CONTROL.

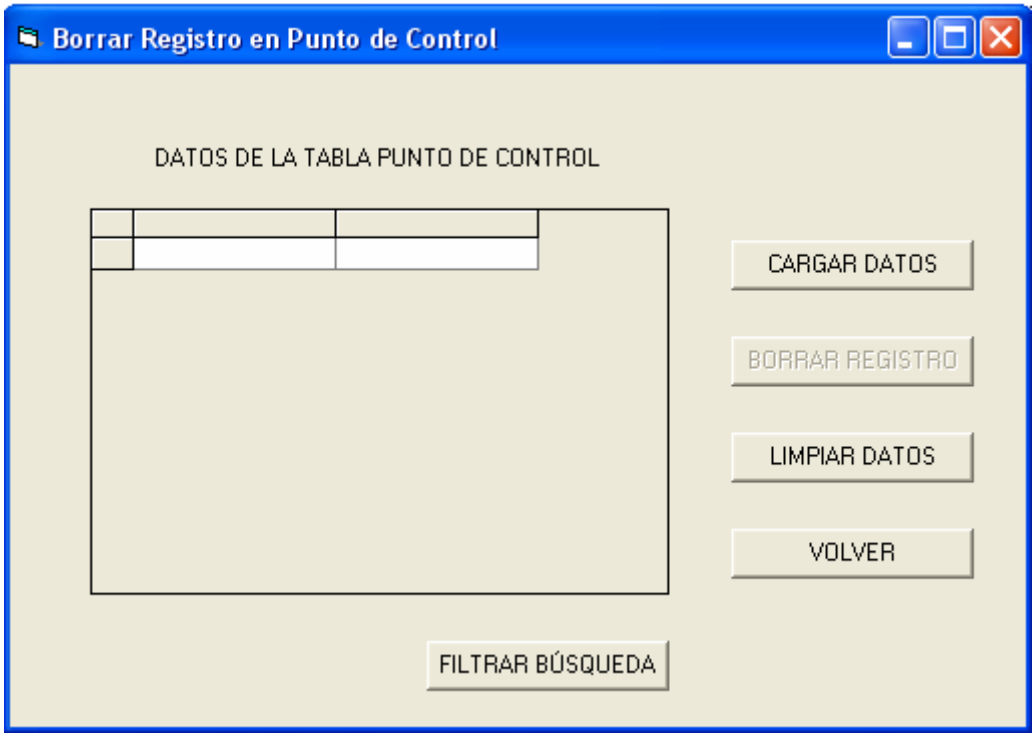

Ilustración 86 : Buscar en Punto Control

En el caso en que no se encuentren datos a mostrar en la tabla de VEHÍCULO se muestra el siguiente mensaje.

Ilustración 87 : Tabla vacía

Si se han encontrado datos, se muestran del modo que indica la siguiente pantalla.

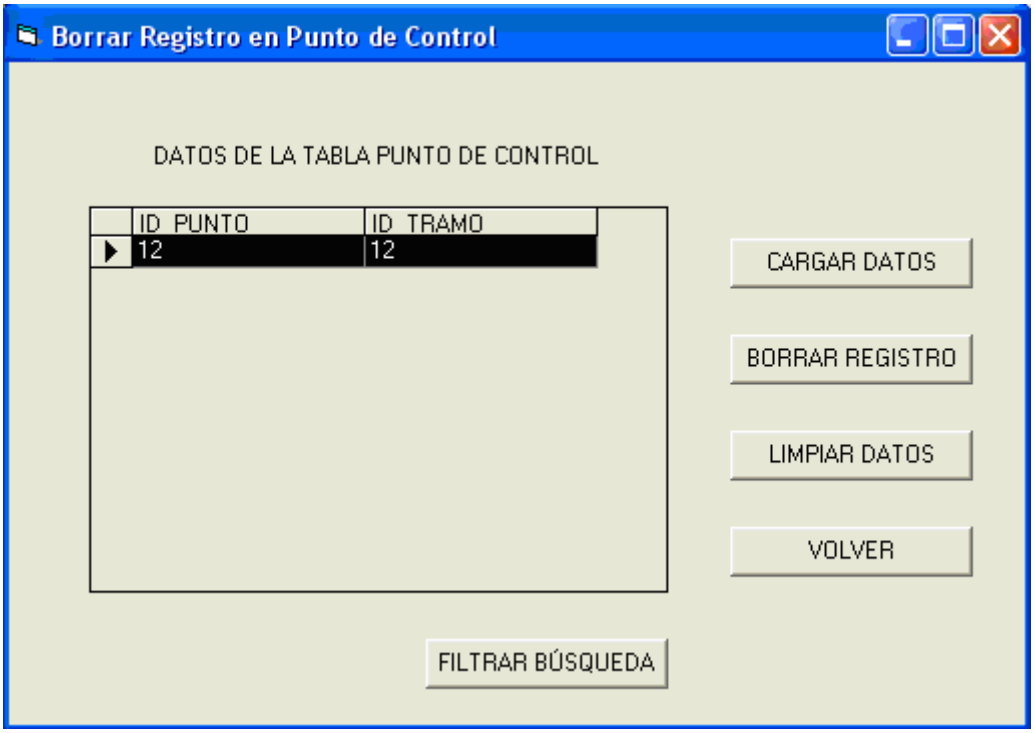

Ilustración 88 : Resultados en Punto Control

La opción de Filtrar Búsqueda muestra las siguientes posibilidades:

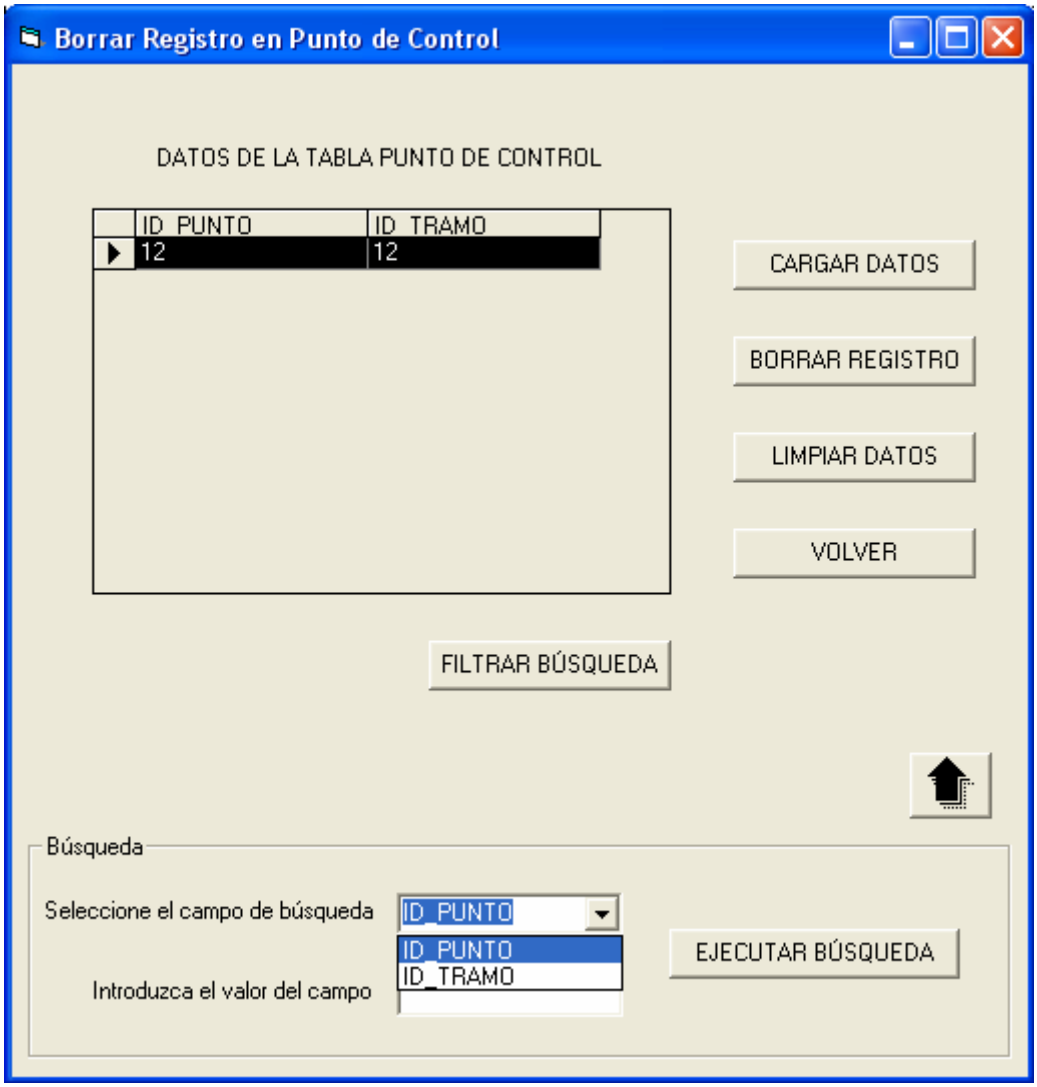

Ilustración 89 : Borrar registro en Punto Control

Una vez seleccionado el registro a eliminar se muestra un mensaje para asegurarse que se quiere eliminar el registro de la base de datos.

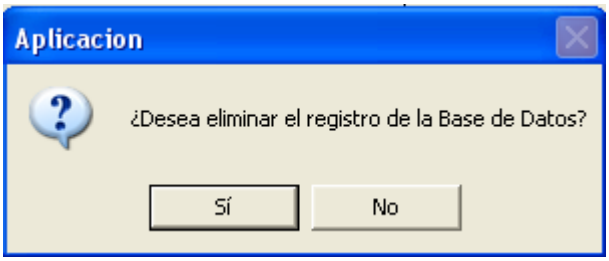

Ilustración 90 : Confirmación borrado

6.3.3 Borrar un tramo

Lo primero que se debe hacer es seleccionar la tabla correspondiente a TRAMO.

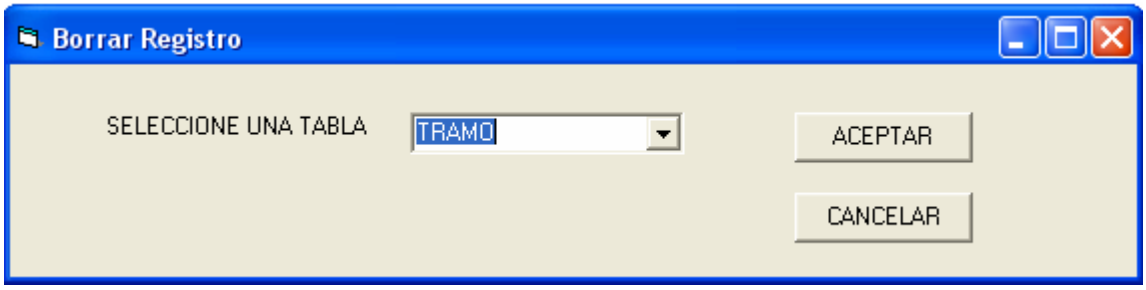

Ilustración 91 : Borrado. Seleccionar tabla Tramo

Una vez seleccionada la tabla, se muestra la siguiente pantalla. Para obtener los datos de la tabla TRAMO, se debe pulsar el botón de Cargar Datos, en el caso en que existan datos dentro de la tabla se cargaran dichos datos en la tabla que aparece. El botón de Limpiar Datos borra de la tabla los datos mostrados, pero no de la base de datos. El botón Volver regresa a la pantalla anterior. Si se pulsa el botón de Filtrar Búsqueda se extiende una parte del formulario que permite filtrar la búsqueda que se quiere realizar por los campos que componen la tabla TRAMO.

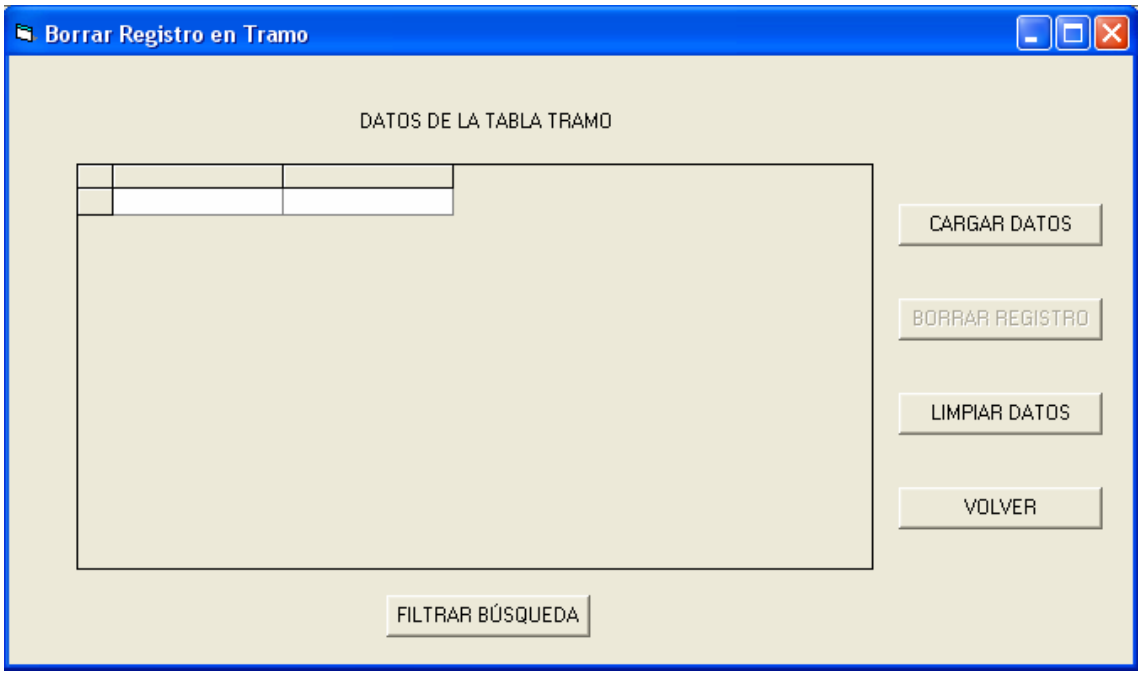

Ilustración 92 : Buscar en Tramo

En el caso en que no se encuentren datos a mostrar en la tabla de TRAMO se muestra el siguiente mensaje.

Ilustración 93 : Tabla vacía

Si se han encontrado datos, se muestran del modo que indica la siguiente pantalla.

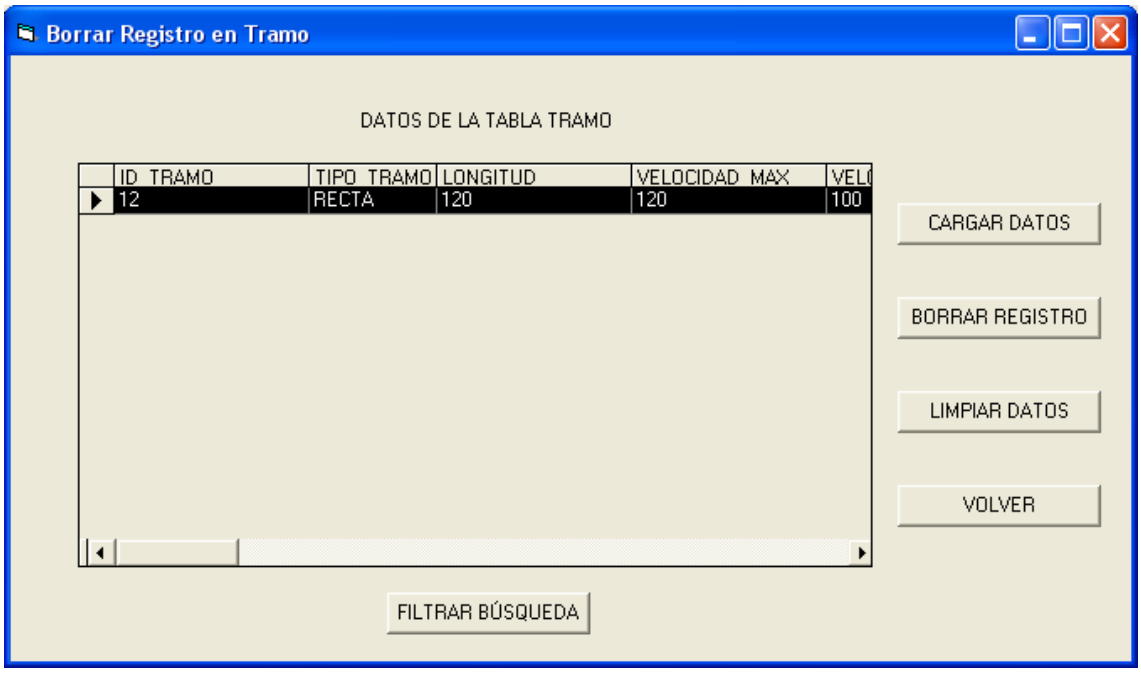

Ilustración 94 : Buscar registro en Tramo

La opción de Filtrar Búsqueda muestra las siguientes posibilidades:

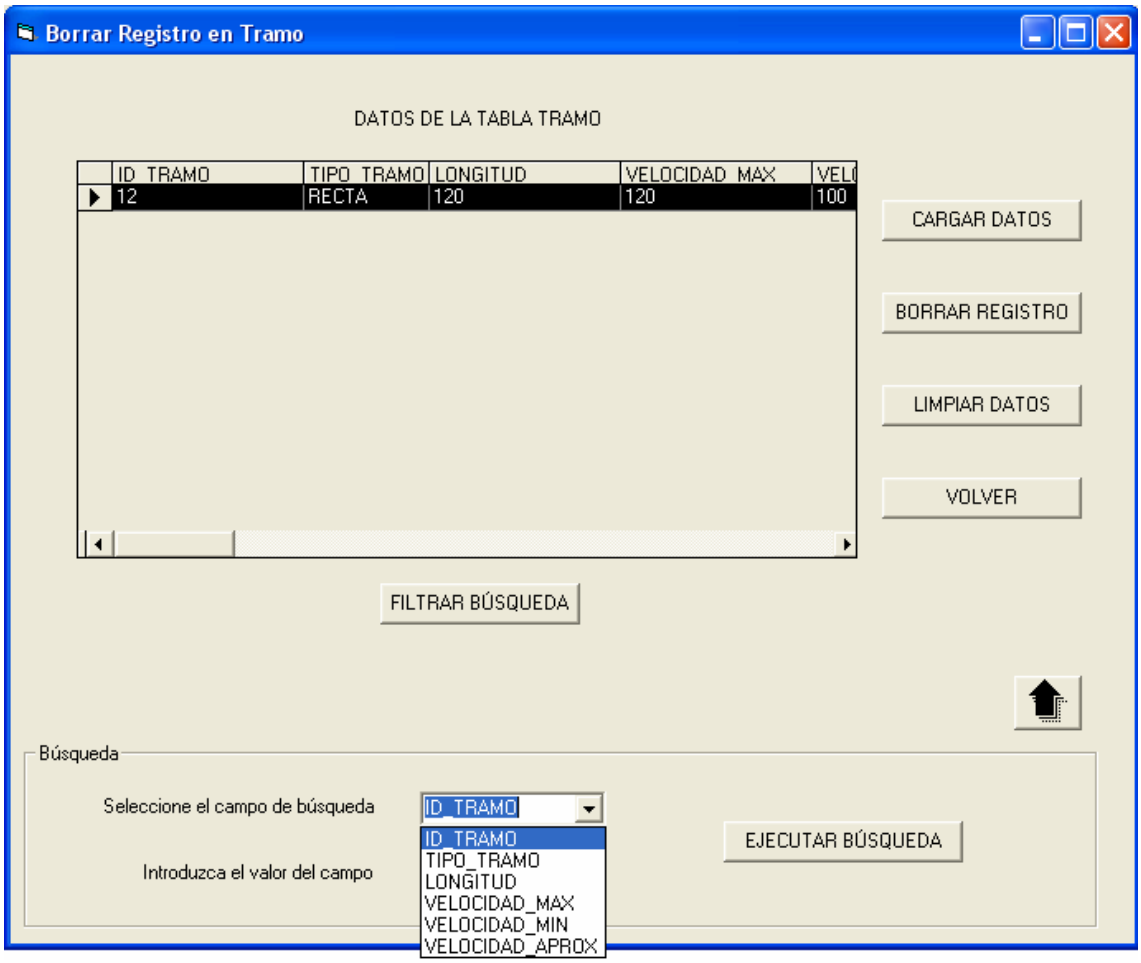

Ilustración 95 : Filtrar búsqueda en Tramo

Una vez seleccionado el registro a eliminar se muestra un mensaje para asegurarse que se quiere eliminar el registro de la base de datos.

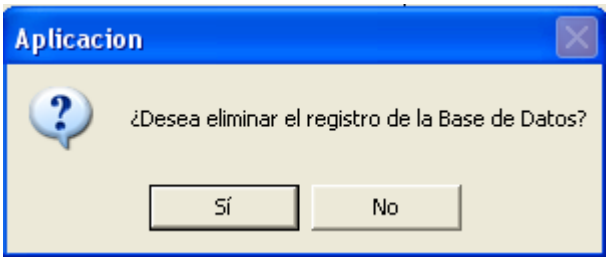

Ilustración 96 : Confirmación borrado

6.3.4 *Borrar un recorrido*

Lo primero que se debe hacer es seleccionar la tabla correspondiente a RECORRIDO.

Ilustración 97 : Borrado. Seleccionar tabla Recorrido

Una vez seleccionada la tabla, se muestra la siguiente pantalla. Para obtener los datos de la tabla RECORRIDO, se debe pulsar el botón de Cargar Datos, en el caso en que existan datos dentro de la tabla se cargaran dichos datos en la tabla que aparece. El botón de Limpiar Datos borra de la tabla los datos mostrados, pero no de la base de datos. El botón Volver regresa a la pantalla anterior. Si se pulsa el botón de Filtrar Búsqueda se extiende una parte del formulario que permite filtrar la búsqueda que se quiere realizar por los campos que componen la tabla RECORRIDO.

Ilustración 98 : Buscar en Recorrido

En el caso en que no se encuentren datos a mostrar en la tabla de RECORRIDO se muestra el siguiente mensaje.

Ilustración 99 : Tabla vacía

Si se han encontrado datos, se muestran del modo que indica la siguiente pantalla.

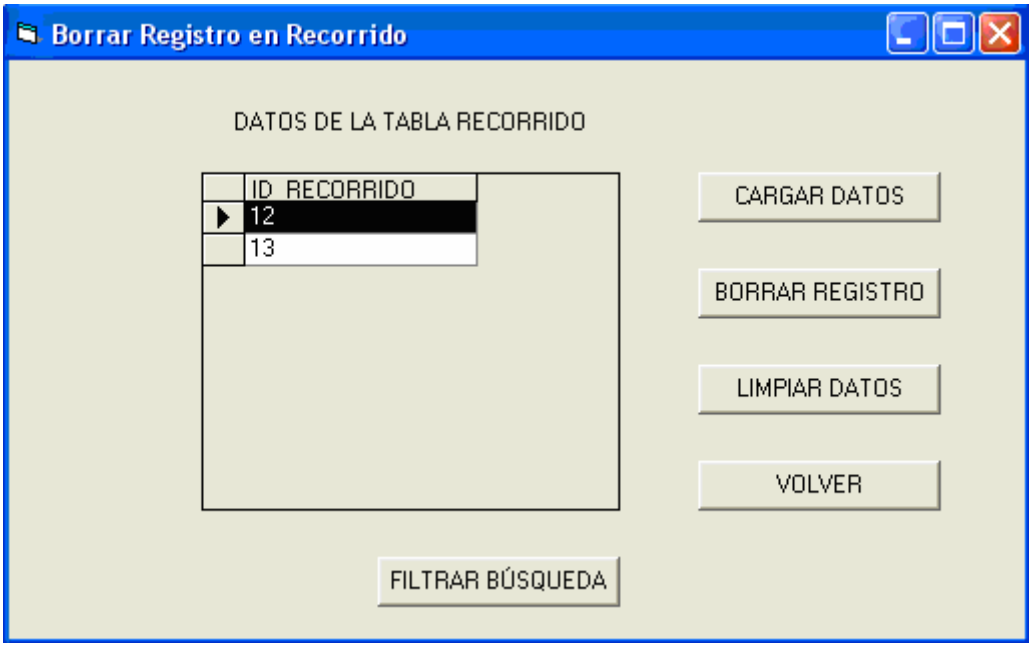

Ilustración 100 : Buscar en Recorrido

La opción de Filtrar Búsqueda muestra las siguientes posibilidades:

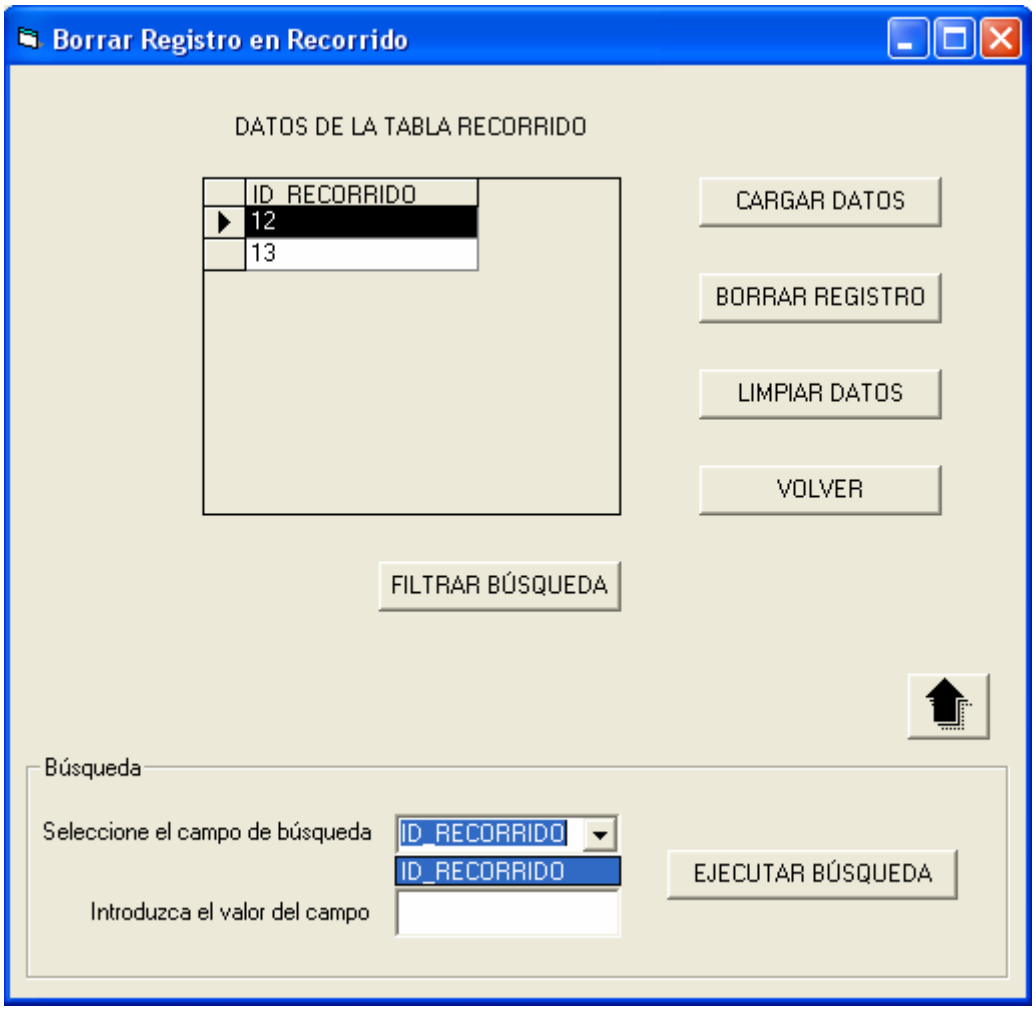

Ilustración 101 : Filtrar búsqueda en Recorrido

Una vez seleccionado el registro a eliminar se muestra un mensaje para asegurarse que se quiere eliminar el registro de la base de datos.

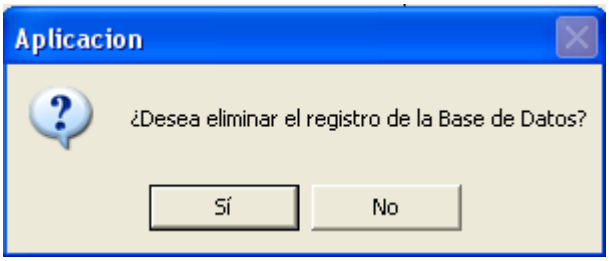

Ilustración 102 : Confirmación borrado

6.3.5 *Borrar un registro de pasa*

Lo primero que se debe hacer es seleccionar la tabla correspondiente a PASA.

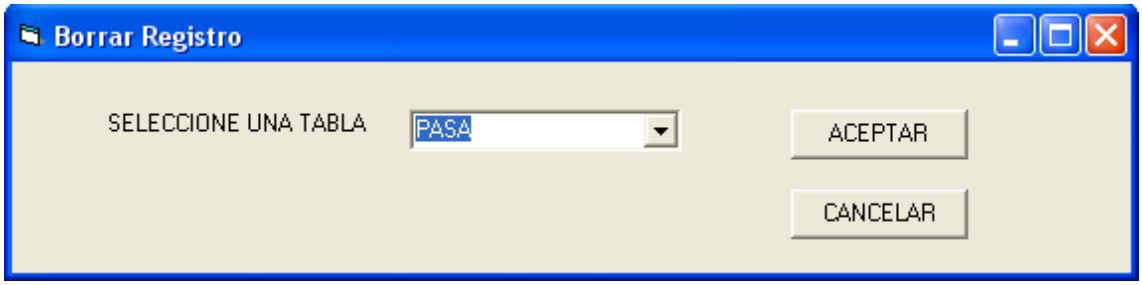

Ilustración 103 : Borrado. Seleccionar tabla Pasa

Una vez seleccionada la tabla, se muestra la siguiente pantalla. Para obtener los datos de la tabla PASA, se debe pulsar el botón de Cargar Datos, en el caso en que existan datos dentro de la tabla se cargaran dichos datos en la tabla que aparece. El botón de Limpiar Datos borra de la tabla los datos mostrados, pero no de la base de datos. El botón Volver regresa a la pantalla anterior. Si se pulsa el botón de Filtrar Búsqueda se extiende una parte del formulario que permite filtrar la búsqueda que se quiere realizar por los campos que componen la tabla PASA.

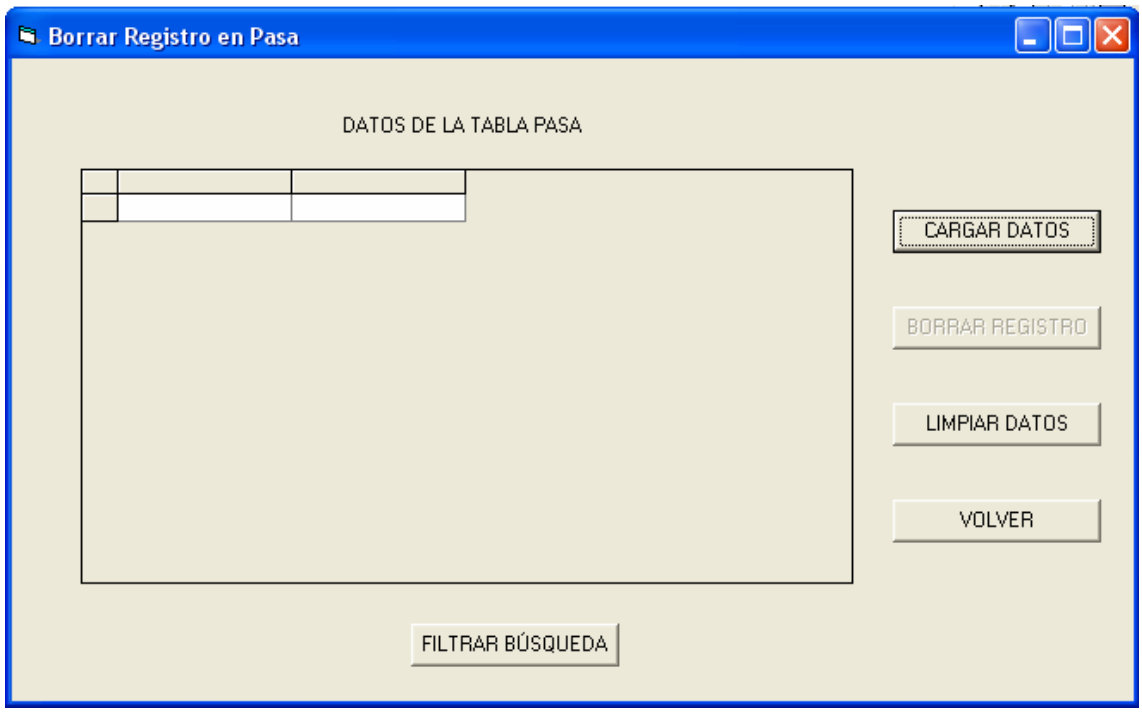

Ilustración 104 : Buscar en Pasa

En el caso en que no se encuentren datos a mostrar en la tabla de PASA se muestra el siguiente mensaje.

Ilustración 105 : Tabla vacía

Si se han encontrado datos, se muestran del modo que indica la siguiente pantalla.

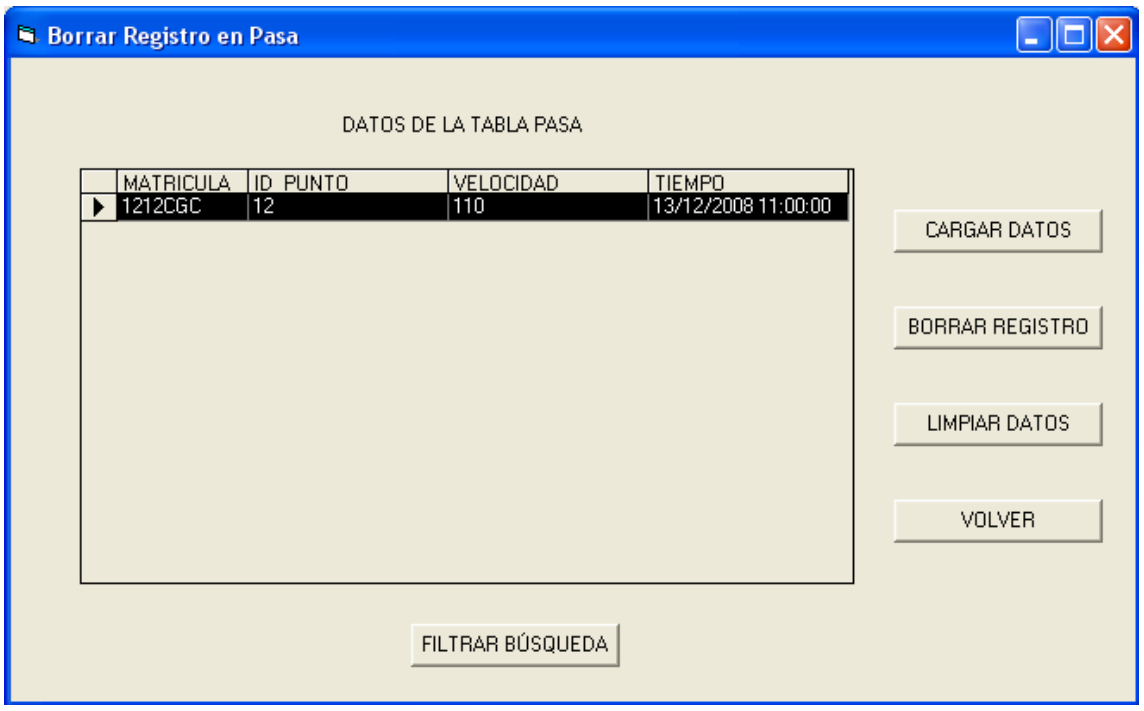

Ilustración 106 : Buscar registro en Pasa

La opción de Filtrar Búsqueda muestra las siguientes posibilidades:

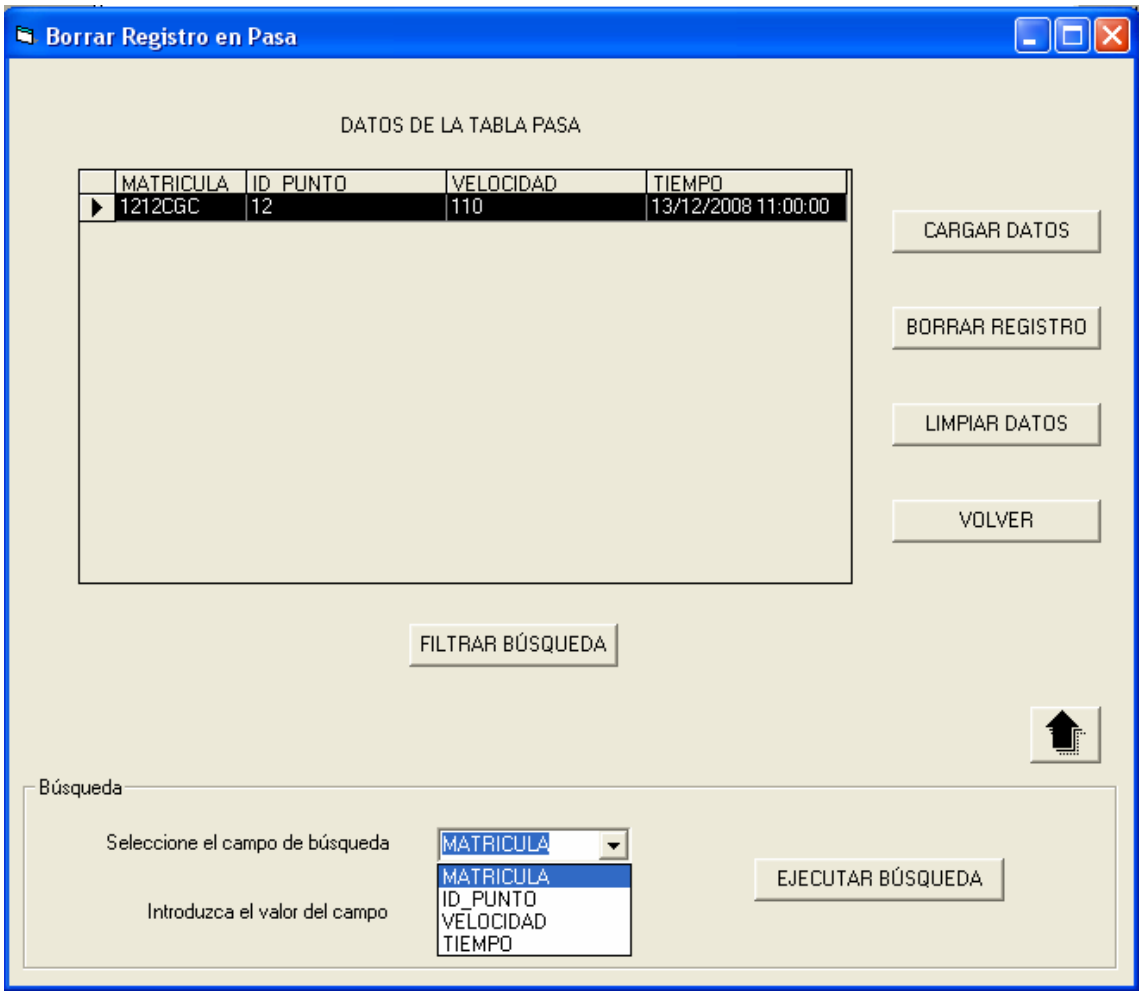

Ilustración 107 : Filtrar búsqueda en Pasa

Una vez seleccionado el registro a eliminar se muestra un mensaje para asegurarse que se quiere eliminar el registro de la base de datos.

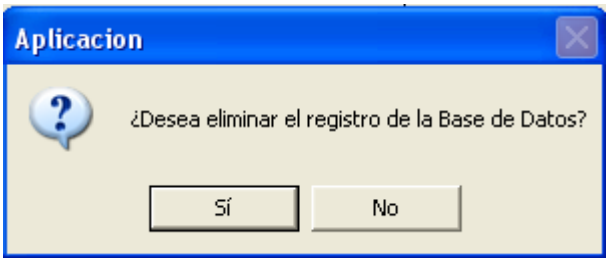

Ilustración 108 : Confirmación borrado

6.3.6 Borrar un registro de pertenece

Lo primero que se debe hacer es seleccionar la tabla correspondiente a PERTENECE.

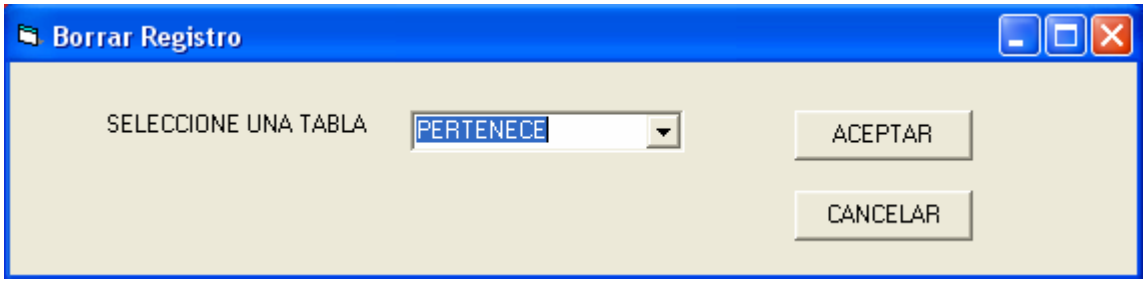

Ilustración 109 : Borrado. Seleccionar tabla Pertenece

Una vez seleccionada la tabla, se muestra la siguiente pantalla. Para obtener los datos de la tabla PERTENECE, se debe pulsar el botón de Cargar Datos, en el caso en que existan datos dentro de la tabla se cargaran dichos datos en la tabla que aparece. El botón de Limpiar Datos borra de la tabla los datos mostrados, pero no de la base de datos. El botón Volver regresa a la pantalla anterior. Si se pulsa el botón de Filtrar Búsqueda se extiende una parte del formulario que permite filtrar la búsqueda que se quiere realizar por los campos que componen la tabla PERTENECE.

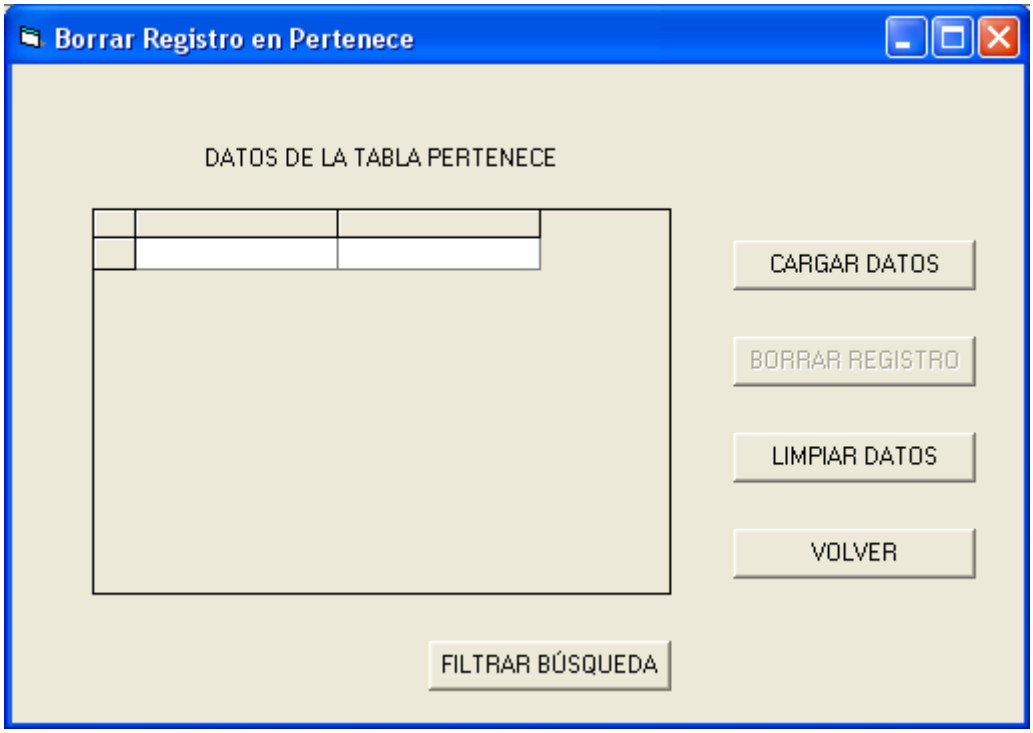

Ilustración 110 : Buscar en Pertenece

En el caso en que no se encuentren datos a mostrar en la tabla de PERTENECE se muestra el siguiente mensaje.

Ilustración 111 : Tabla vacía

Si se han encontrado datos, se muestran del modo que indica la siguiente pantalla.

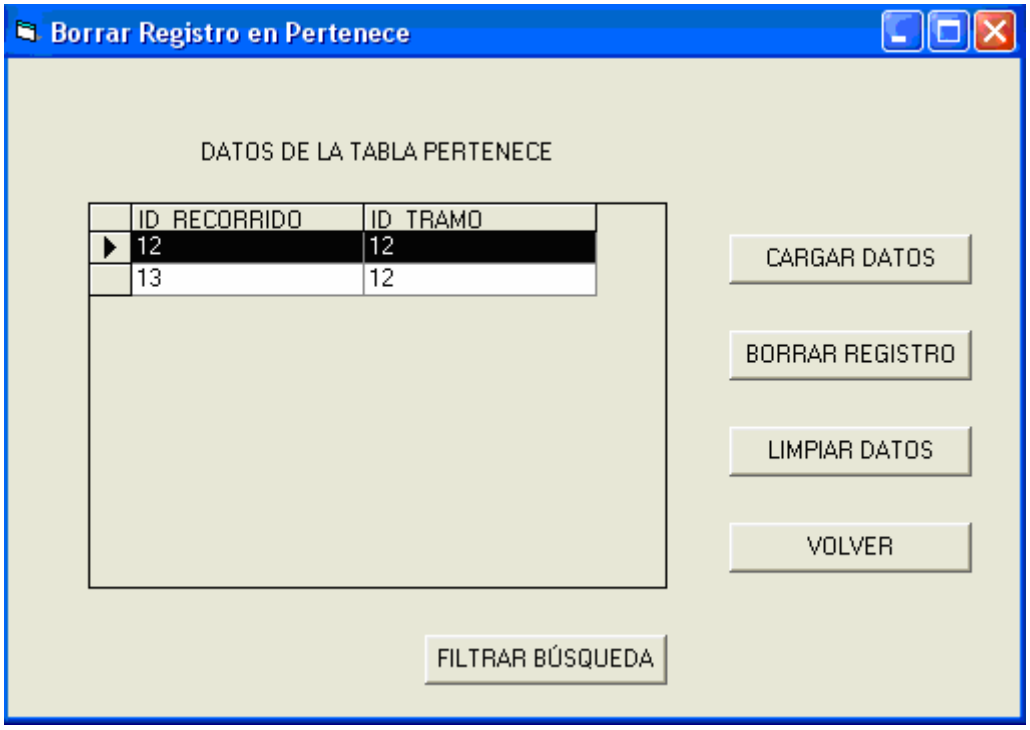

Ilustración 112 : Buscar registro en Pertenece

La opción de Filtrar Búsqueda muestra las siguientes posibilidades:

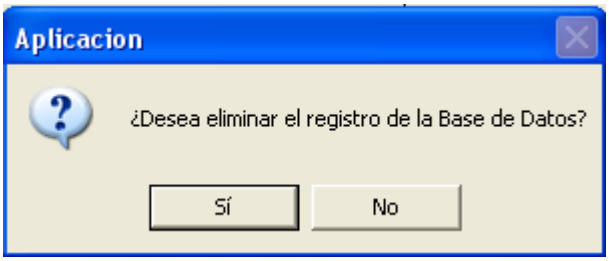

Ilustración 113 : Confirmación borrado

Una vez seleccionado el registro a eliminar se muestra un mensaje para asegurarse que se quiere eliminar el registro de la base de datos.

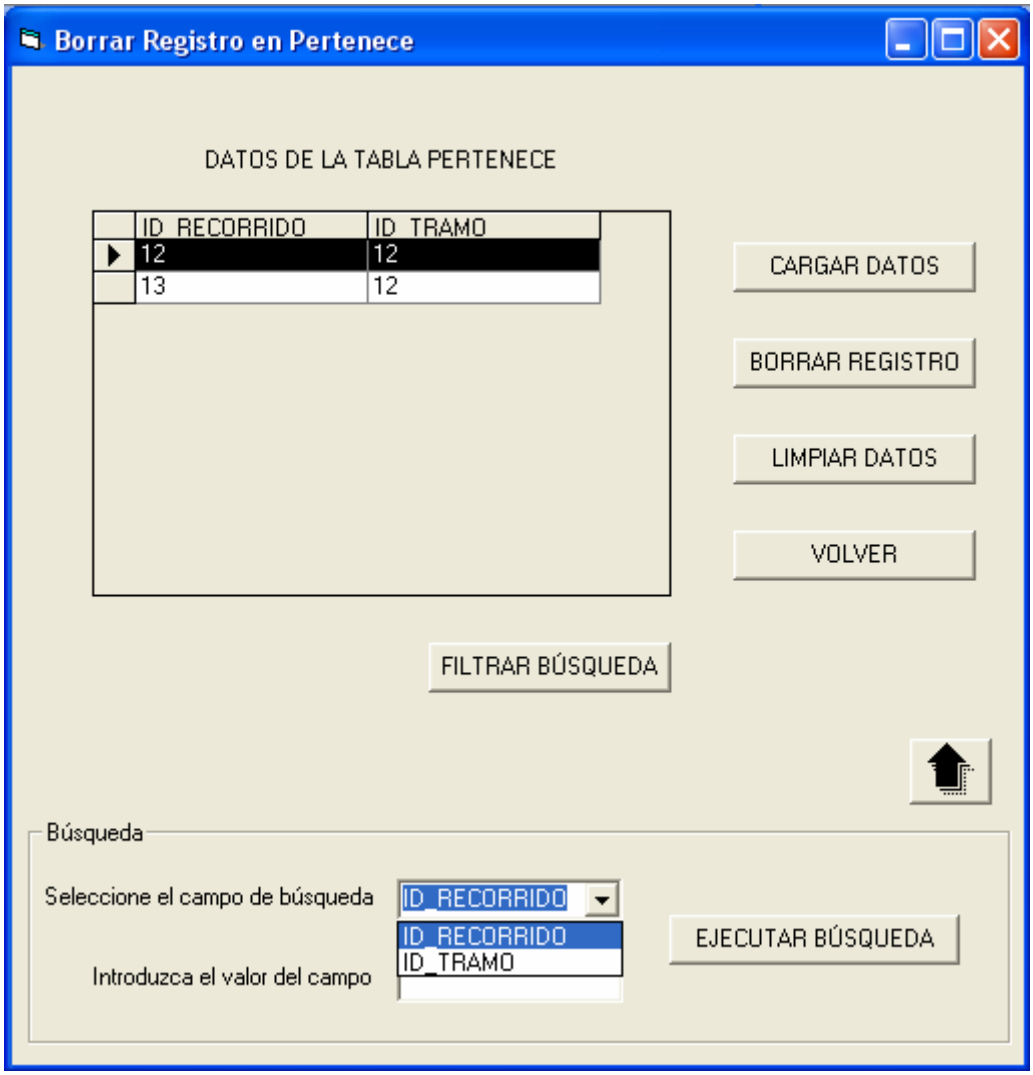

Ilustración 114 : Filtrar búsqueda en Pertenece

6.3.7 Borrar un registro de realiza

Lo primero que se debe hacer es seleccionar la tabla correspondiente a REALIZA.

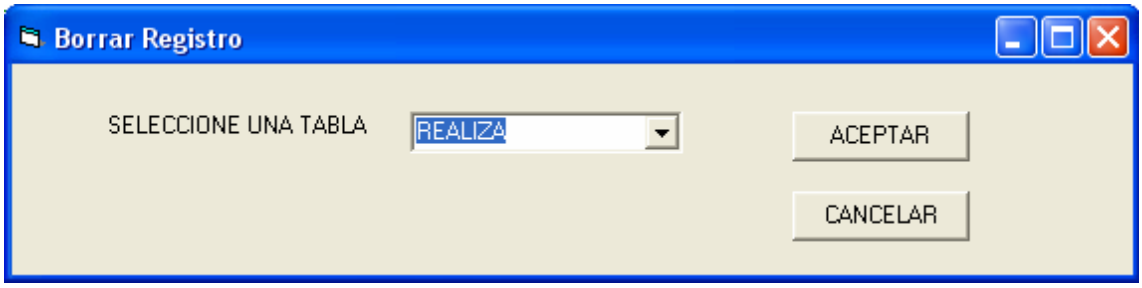

Ilustración 115 : Borrado. Seleccionar tabla Realiza

Una vez seleccionada la tabla, se muestra la siguiente pantalla. Para obtener los datos de la tabla REALIZA, se debe pulsar el botón de Cargar Datos, en el caso en que existan datos dentro de la tabla se cargaran dichos datos en la tabla que aparece. El botón de Limpiar Datos borra de la tabla los datos mostrados, pero no de la base de datos. El botón Volver regresa a la pantalla anterior. Si se pulsa el botón de Filtrar Búsqueda se extiende una parte del formulario que permite filtrar la búsqueda que se quiere realizar por los campos que componen la tabla REALIZA.

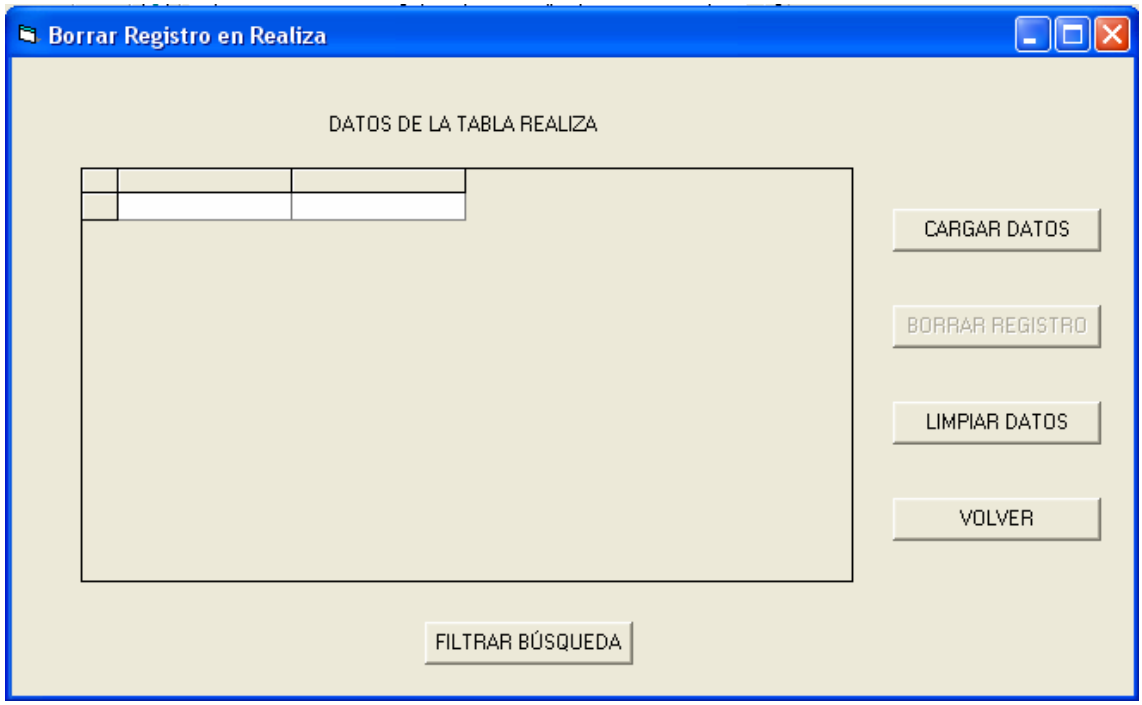

Ilustración 116 : Buscar en Realiza

En el caso en que no se encuentren datos a mostrar en la tabla de REALIZA se muestra el siguiente mensaje.

Ilustración 117 : Tabla vacía

Si se han encontrado datos, se muestran del modo que indica la siguiente pantalla.

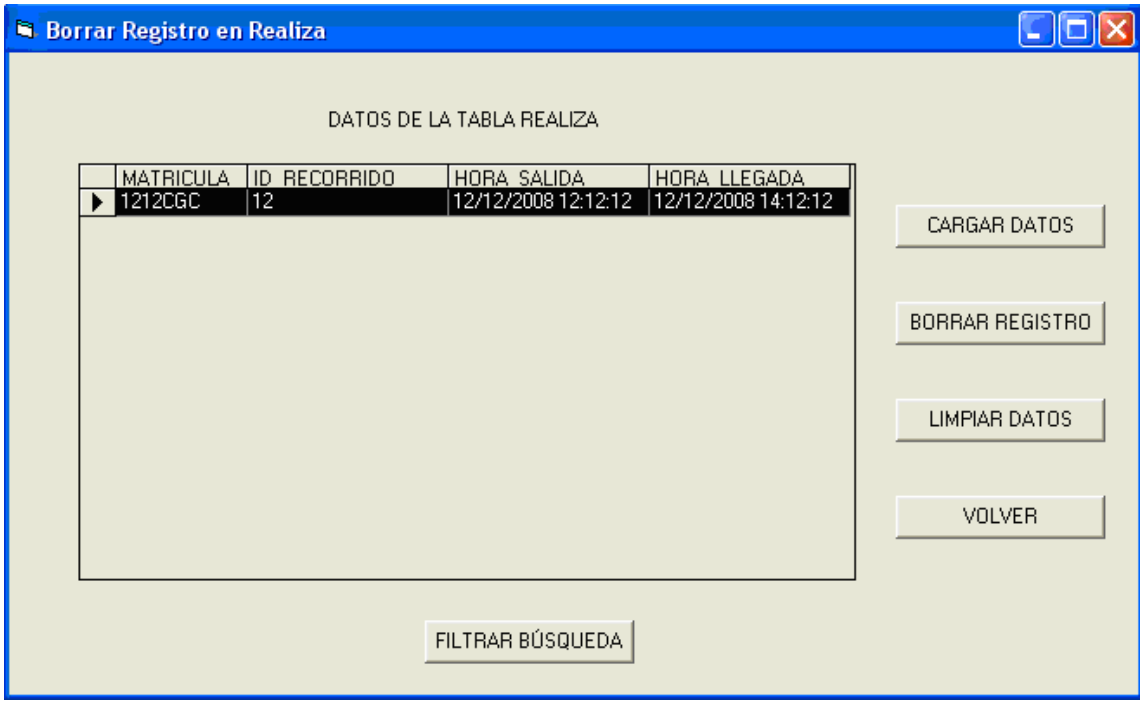

Ilustración 118 : Buscar registro en Realiza

La opción de Filtrar Búsqueda muestra las siguientes posibilidades:

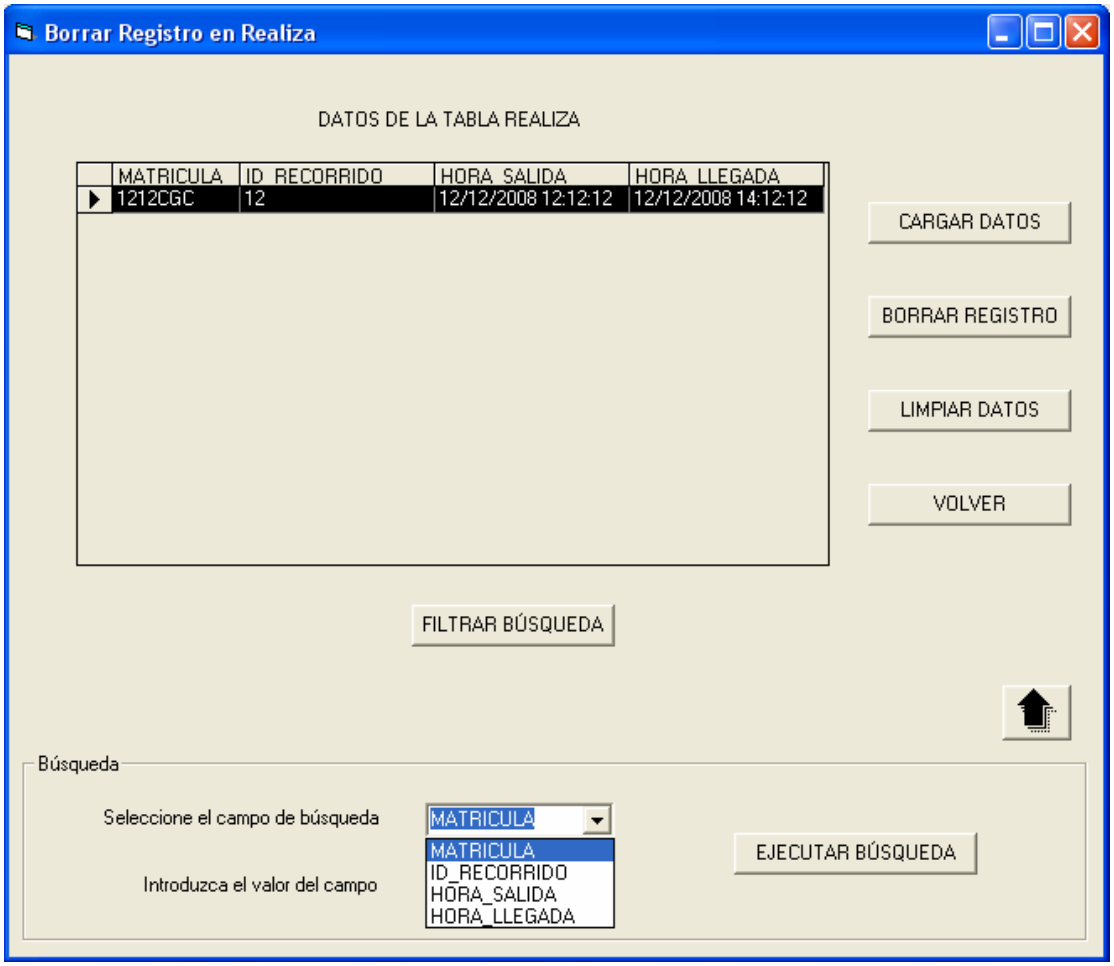

Ilustración 119 : Filtrar búsqueda en Realiza

Una vez seleccionado el registro a eliminar se muestra un mensaje para asegurarse que se quiere eliminar el registro de la base de datos.

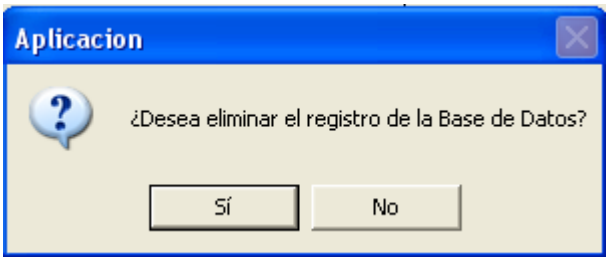

Ilustración 120 : Confirmación borrado

6.4 Pruebas de consulta de datos

En esta sección se muestran los registros de la base de datos a partir de consultas en SQL. Se dispone de siete consultas distintas. Cada una de ellas ofrece una visión diferente de las relaciones entre vehículos y los recorridos que realizan.

6.4.1 Mismo recorrido

En la consulta *Mismo recorrido* se ofrece la posibilidad de acceder a los vehículos que realizan un mismo recorrido. No se especifica el recorrido, por lo que se mostraran todos los que se encuentren en la base de datos.

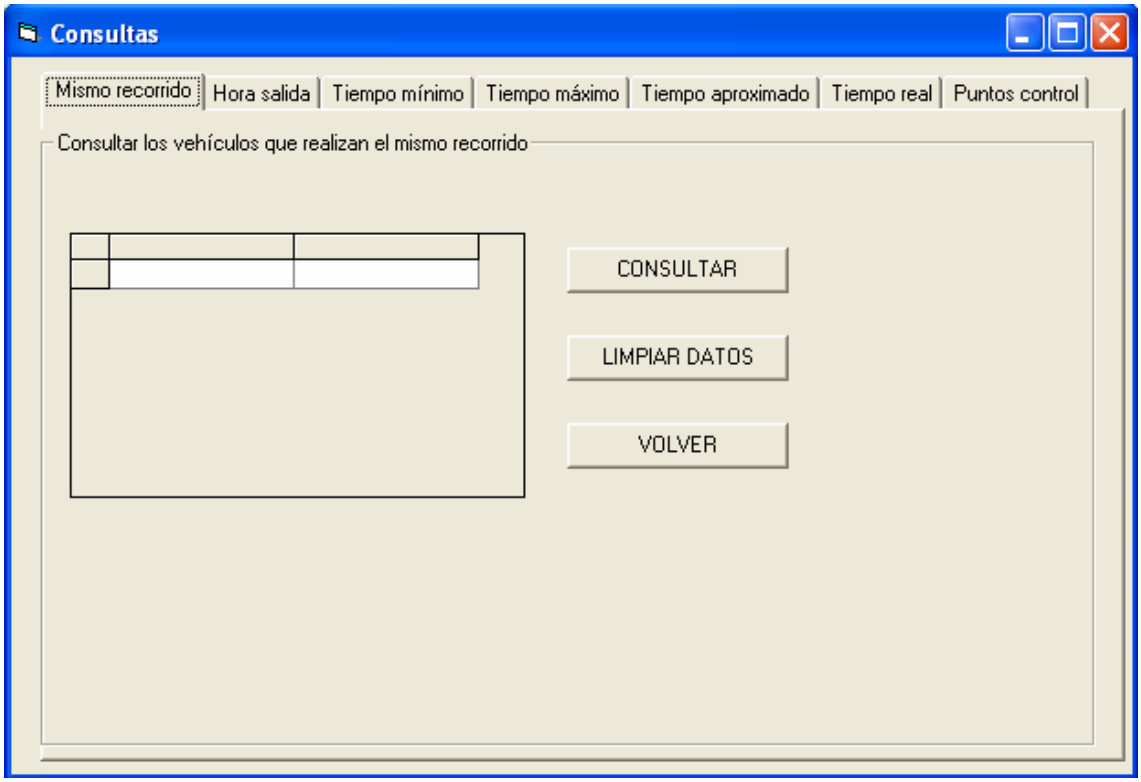

Ilustración 121 : Consulta Mismo recorrido

La siguiente pantalla muestra los datos obtenidos a partir de la consulta lanzada.

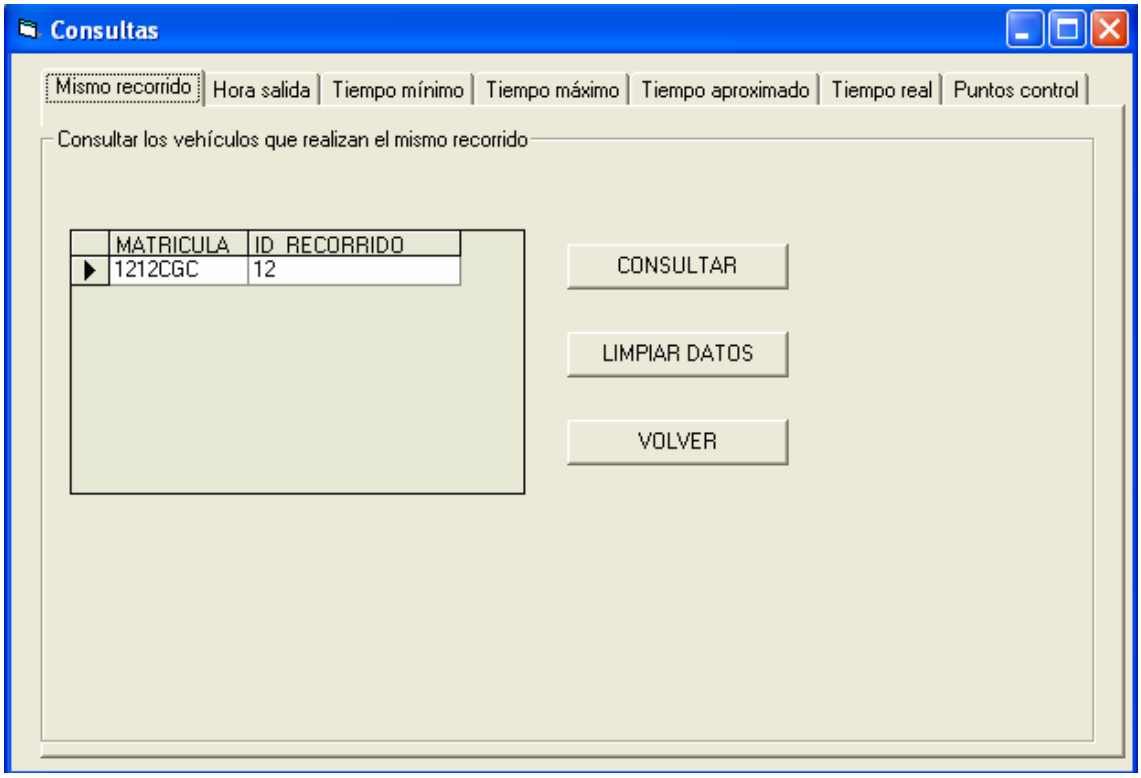

Ilustración 122 : Resultados consulta Mismo recorrido

6.4.2 Hora salida

En la consulta *Hora salida* se ofrece la posibilidad de acceder a las distintas horas de salidas relacionadas con un vehículo en determinado para un recorrido en concreto. Para realizar esta consulta se deben insertar los datos correspondientes a la matrícula del vehículo y al identificador del recorrido a consultar.

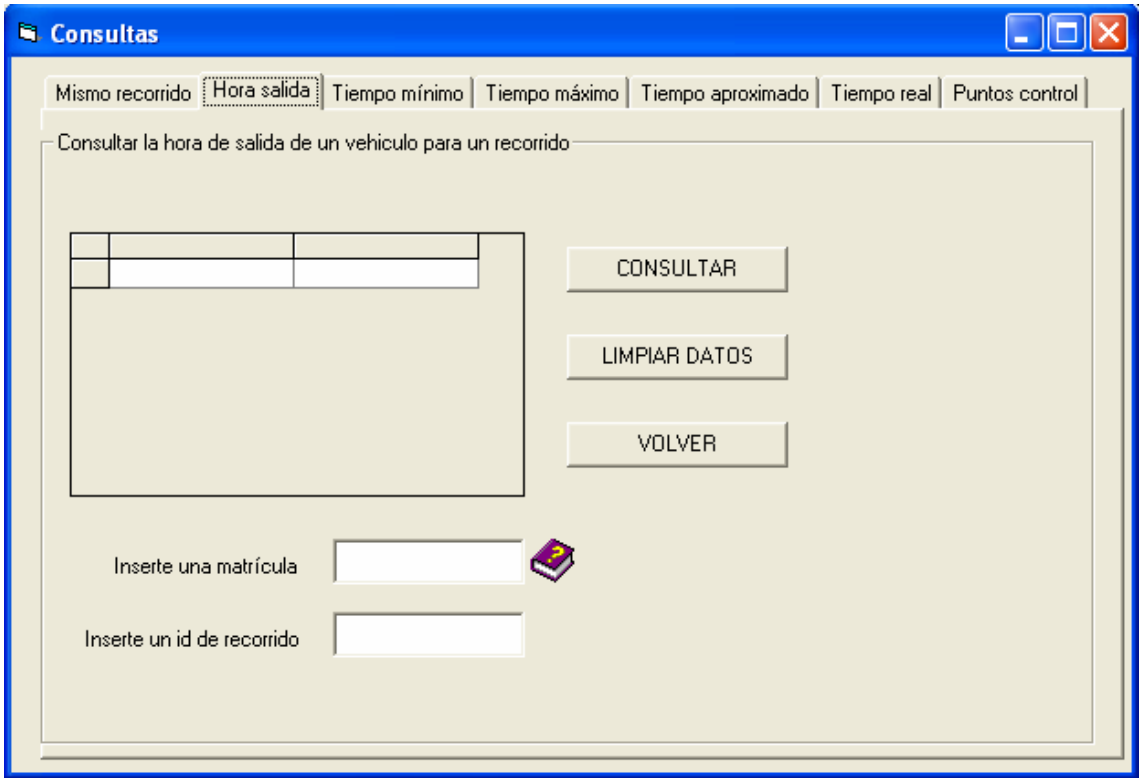

Ilustración 123 : Consulta Hora salida

La siguiente pantalla muestra los datos obtenidos a partir de la consulta lanzada. Para este caso se ha introducido una matricula válida y un recorrido válido.

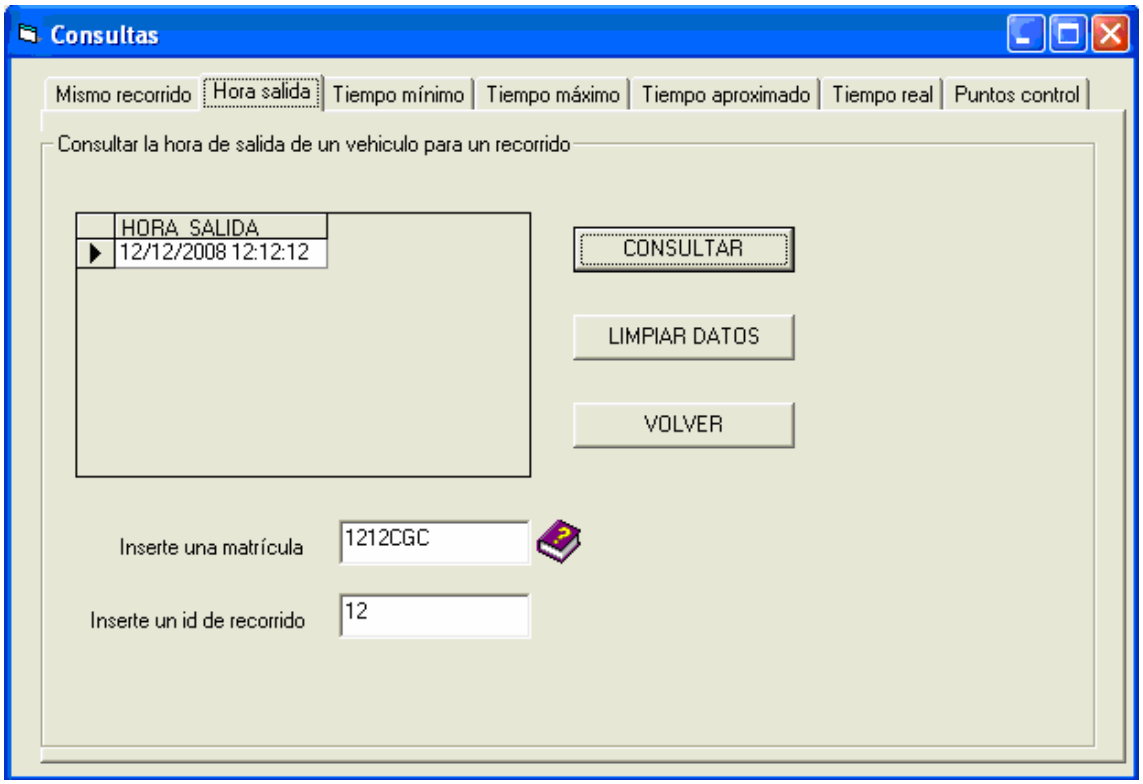

Ilustración 124 : Resultados consulta Hora salida

En el caso en que no se encuentren datos que correspondan con los datos introducidos se mostrara la siguiente pantalla.

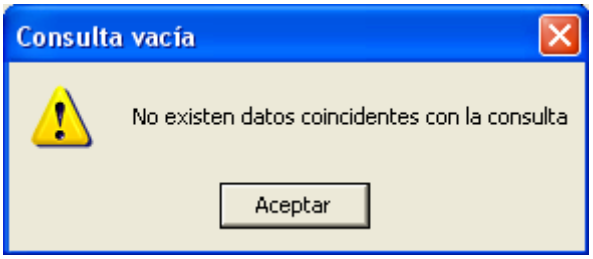

Ilustración 125 : Consulta vacía

6.4.3 Tiempo mínimo

En la consulta *Tiempo mínimo* se ofrece la posibilidad de acceder al tiempo mínimo que un vehículo en concreto tarda en realizar un recorrido determinado. Para poder realizar esta consulta se deberán insertar los datos correspondientes a la matrícula del vehículo, el identificador del recorrido y la hora de salida en que se comenzó el recorrido. Estos tres datos son necesarios ya que conforman la clave primaria de la tabla que registra los recorridos realizados por un vehículo en concreto.

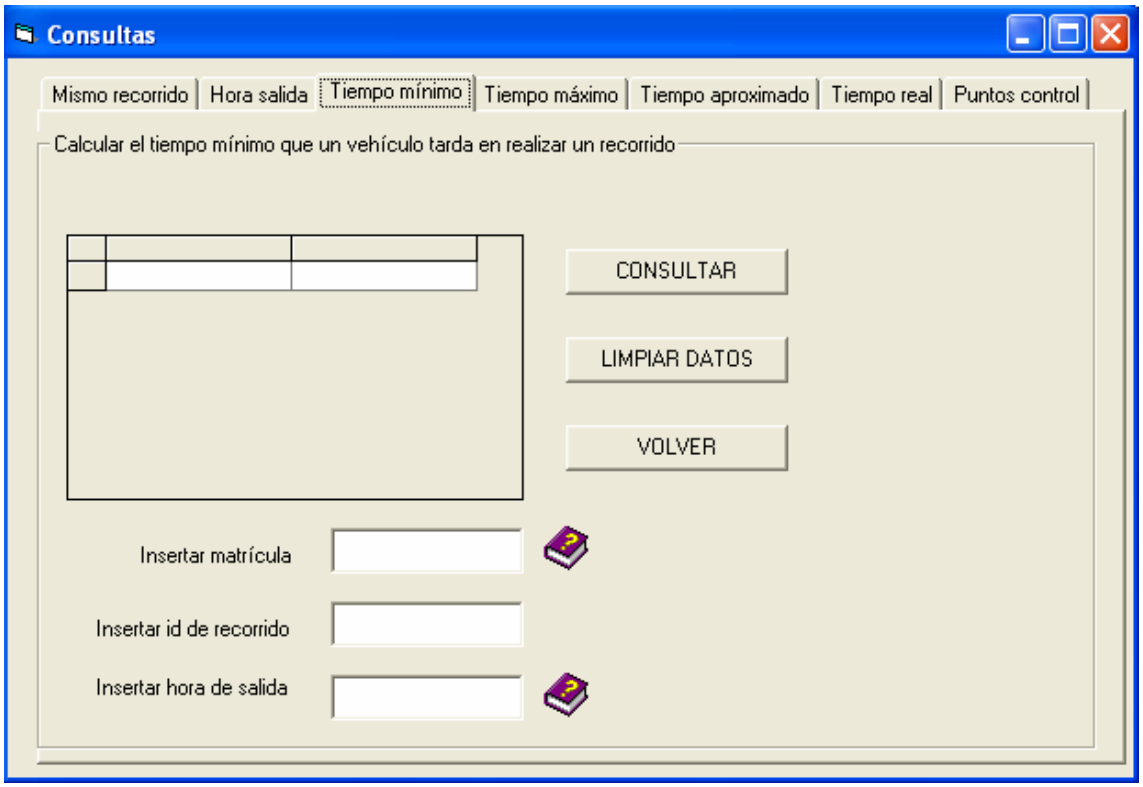

Ilustración 126 : Consulta Tiempo mínimo

La siguiente pantalla muestra los datos obtenidos a partir de la consulta lanzada. Para este caso se ha introducido una matricula válida, un recorrido válido y una hora de salida válida.

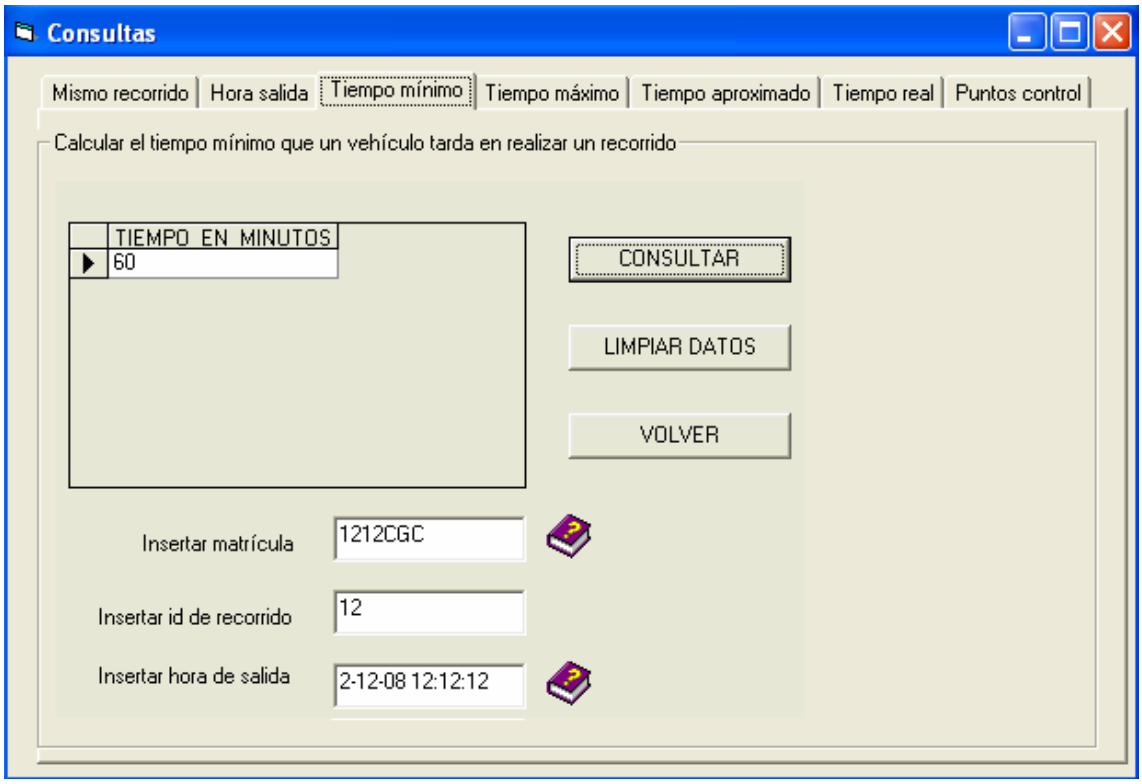

Ilustración 127 : Resultados consulta Tiempo mínimo

En el caso en que no se encuentren datos que correspondan con los datos introducidos se mostrara la siguiente pantalla.

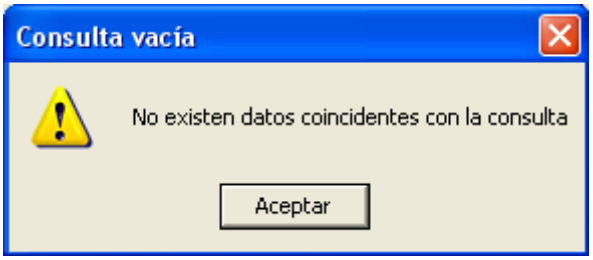

Ilustración 128 : Consulta vacía

6.4.4 *Tiempo máximo*

En la consulta *Tiempo máximo* se ofrece la posibilidad de acceder al tiempo máximo que un vehículo en concreto tarda en realizar un recorrido determinado. Para poder realizar esta consulta se deberán insertar los datos correspondientes a la matrícula del vehículo, el identificador del recorrido y la hora de salida en que se comenzó el recorrido. Estos tres datos son necesarios ya que conforman la clave primaria de la tabla que registra los recorridos realizados por un vehículo en concreto.

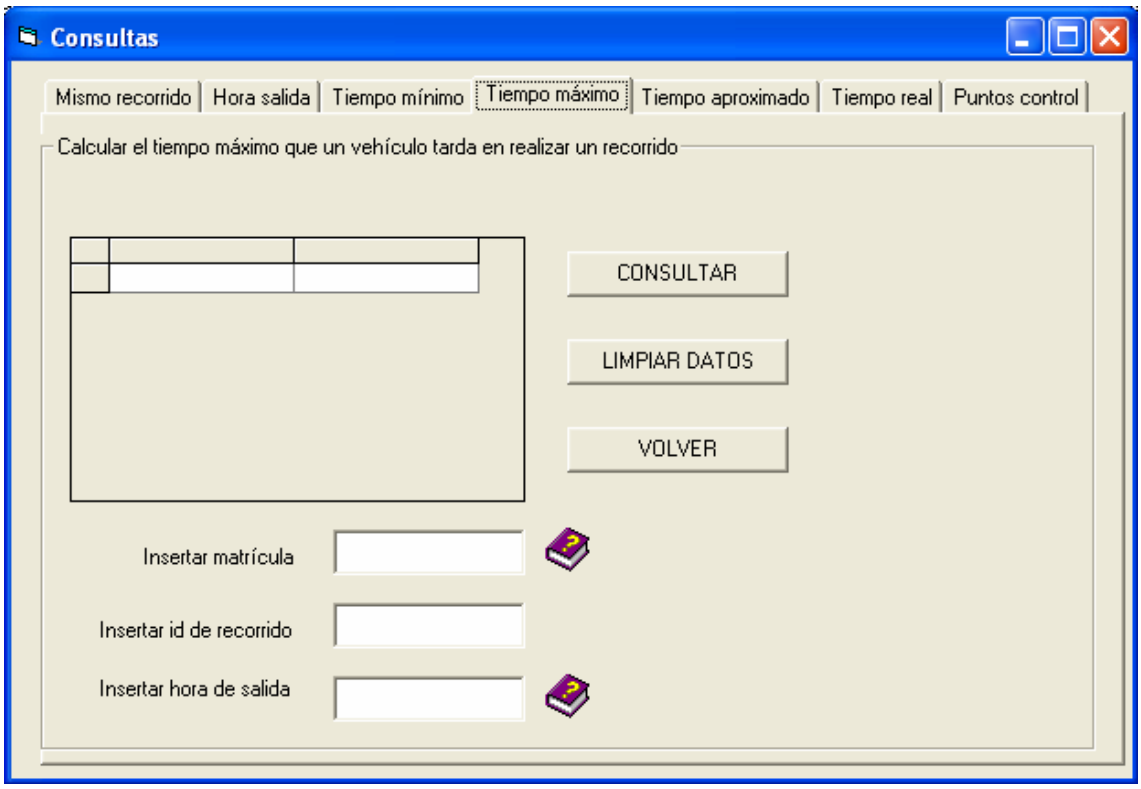

Ilustración 129 : Consulta Tiempo máximo

La siguiente pantalla muestra los datos obtenidos a partir de la consulta lanzada. Para este caso se ha introducido una matricula válida, un recorrido válido y una hora de salida válida.

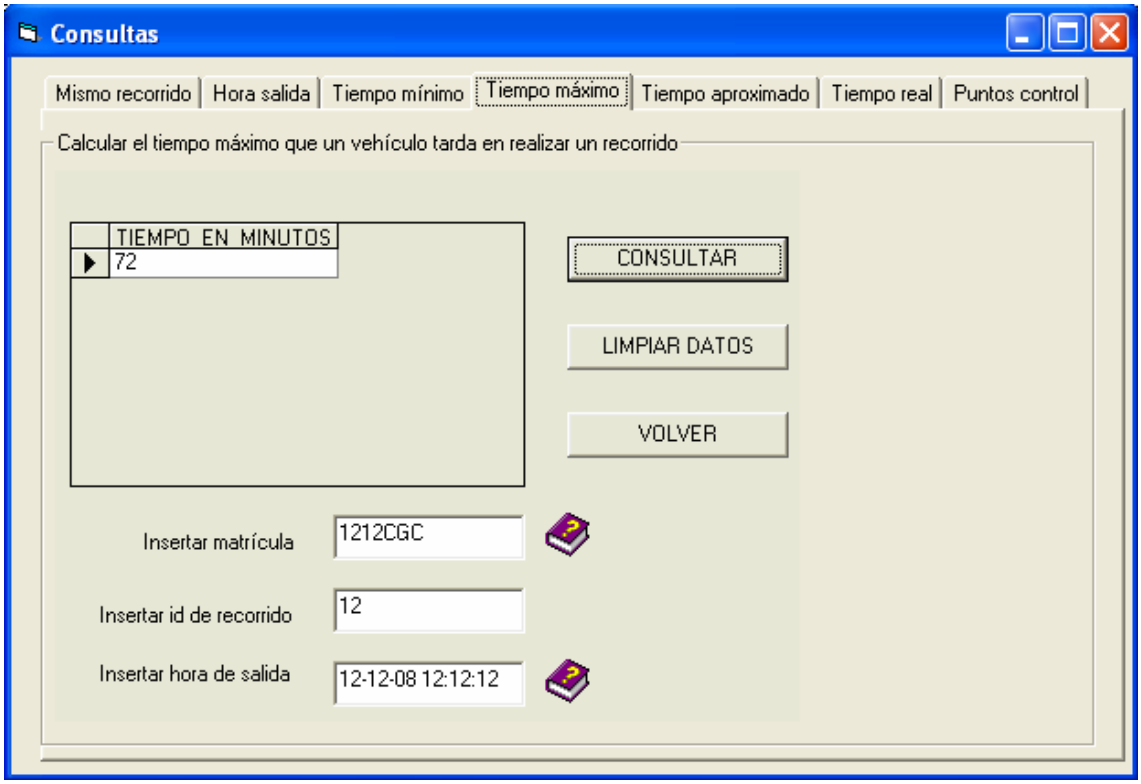

Ilustración 130 : Resultados consulta Tiempo máximo

En el caso en que no se encuentren datos que correspondan con los datos introducidos se mostrara la siguiente pantalla.

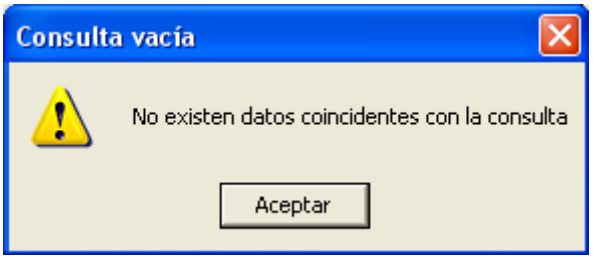

Ilustración 131 : Consulta vacía

6.4.5 Tiempo aproximado

En la consulta *Tiempo aproximado* se ofrece la posibilidad de acceder al tiempo aproximado que un vehículo en concreto tarda en realizar un recorrido determinado. Para poder realizar esta consulta se deberán insertar los datos correspondientes a la matrícula del vehículo, el identificador del recorrido y la hora de salida en que se comenzó el recorrido. Estos tres datos son necesarios ya que conforman la clave primaria de la tabla que registra los recorridos realizados por un vehículo en concreto.

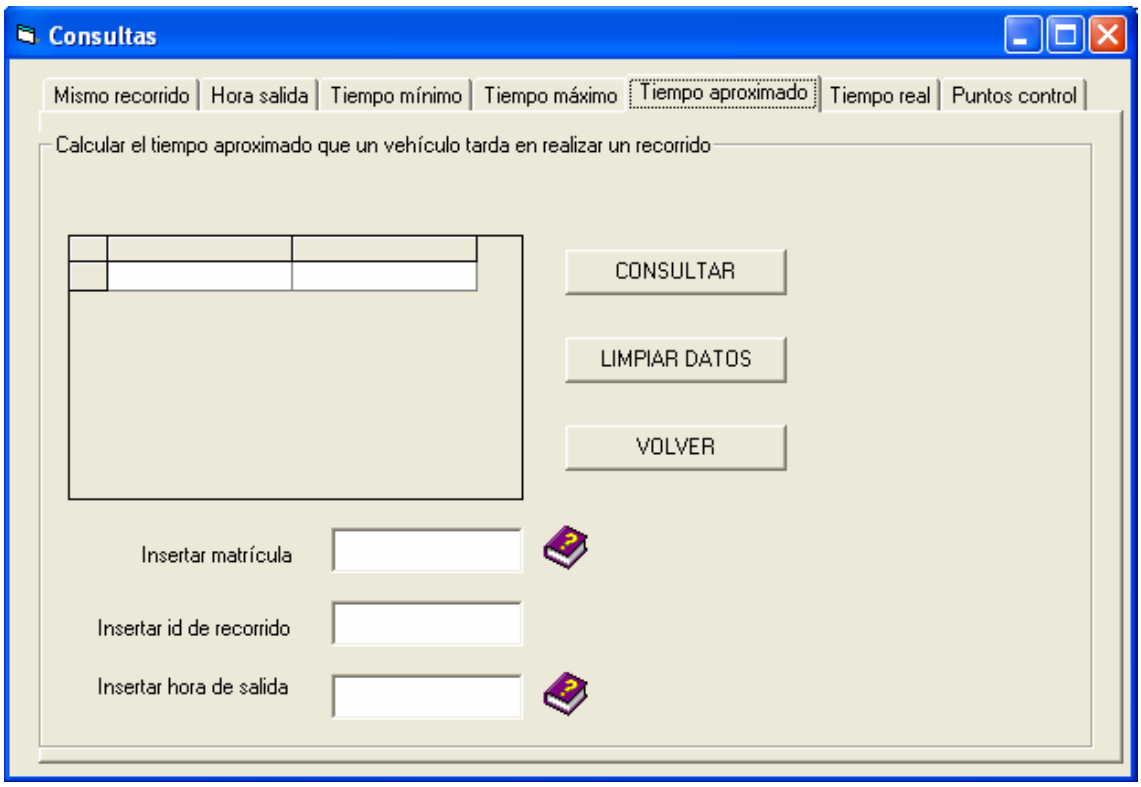

Ilustración 132 : Consulta Tiempo aproximado

La siguiente pantalla muestra los datos obtenidos a partir de la consulta lanzada. Para este caso se ha introducido una matricula válida, un recorrido válido y una hora de salida válida.

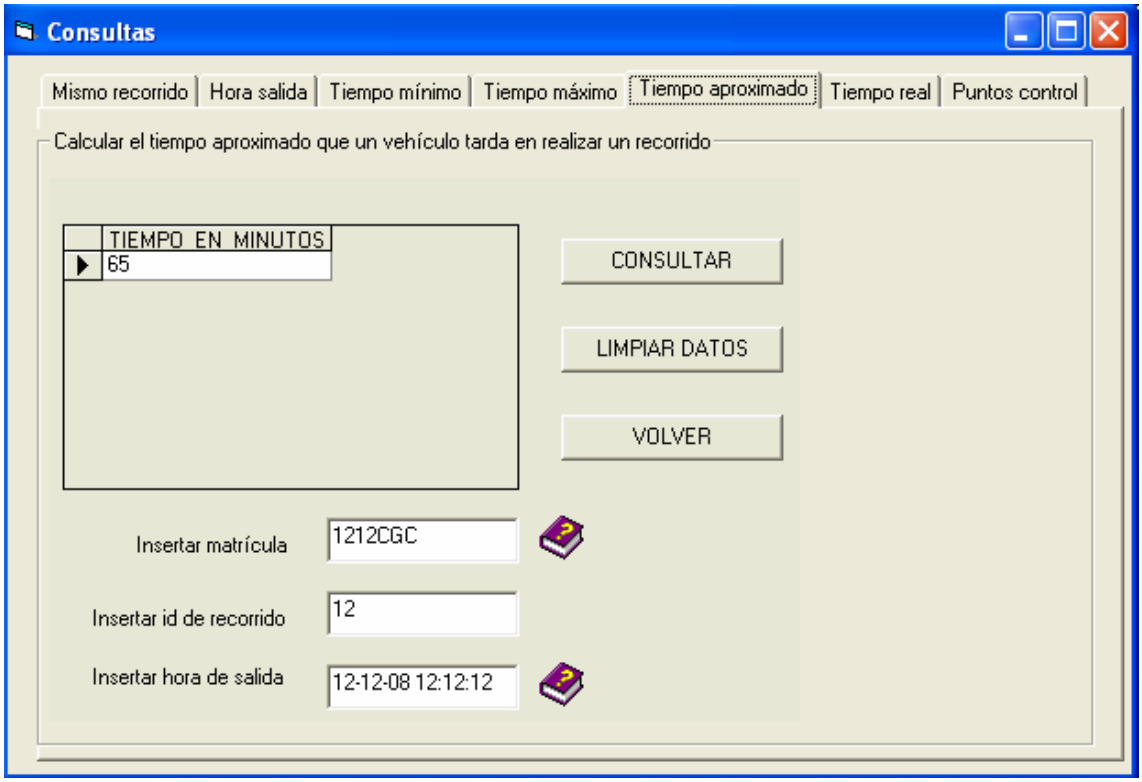

Ilustración 133 : Resultados consulta Tiempo aproximado

En el caso en que no se encuentren datos que correspondan con los datos introducidos se mostrara la siguiente pantalla.

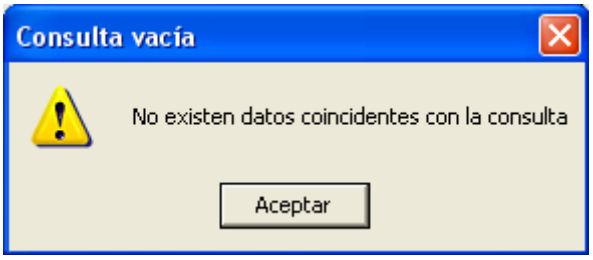

Ilustración 134 : Consulta vacía

6.4.6 *Tiempo real*

En la consulta *Tiempo real* se ofrece la posibilidad de acceder al tiempo aproximado que un vehículo en concreto tarda en realizar un recorrido determinado. Para poder realizar esta consulta se deberán insertar los datos correspondientes a la matrícula del vehículo, el identificador del recorrido y la hora de salida en que se comenzó el recorrido. Estos tres datos son necesarios ya que conforman la clave primaria de la tabla que registra los recorridos realizados por un vehículo en concreto.

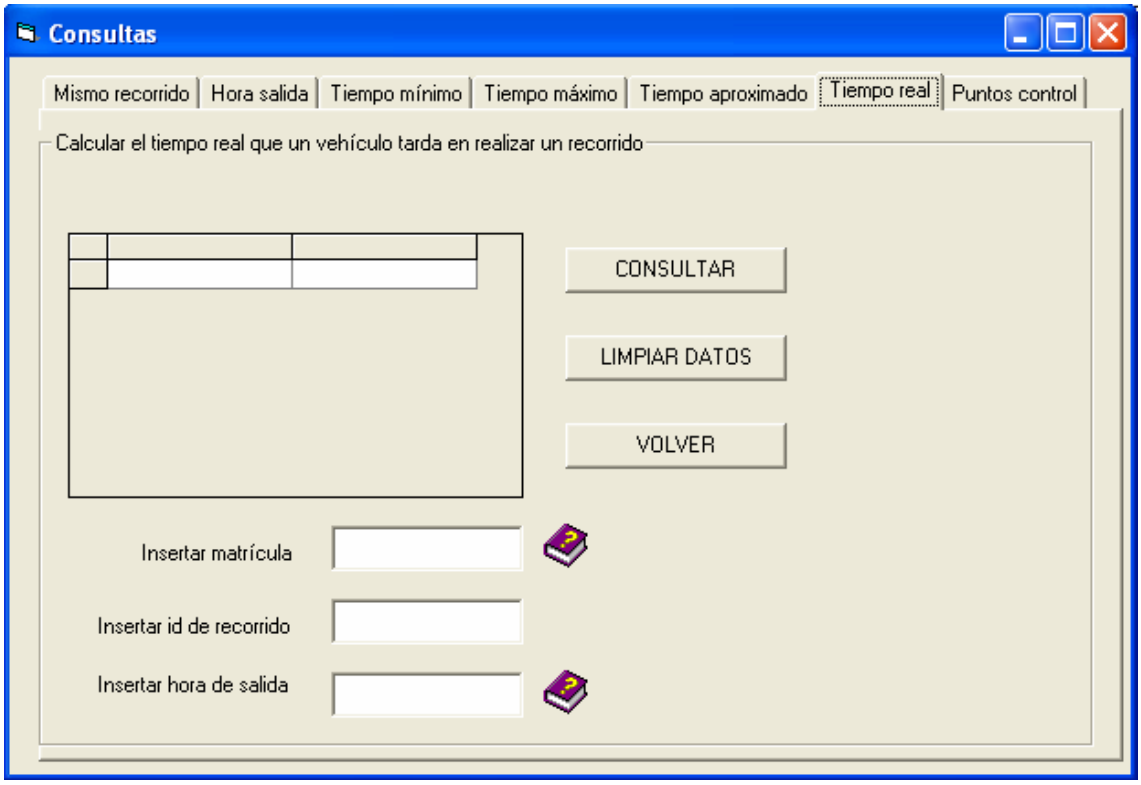

Ilustración 135 : Consulta Tiempo real

La siguiente pantalla muestra los datos obtenidos a partir de la consulta lanzada. Para este caso se ha introducido una matricula válida, un recorrido válido y una hora de salida válida.

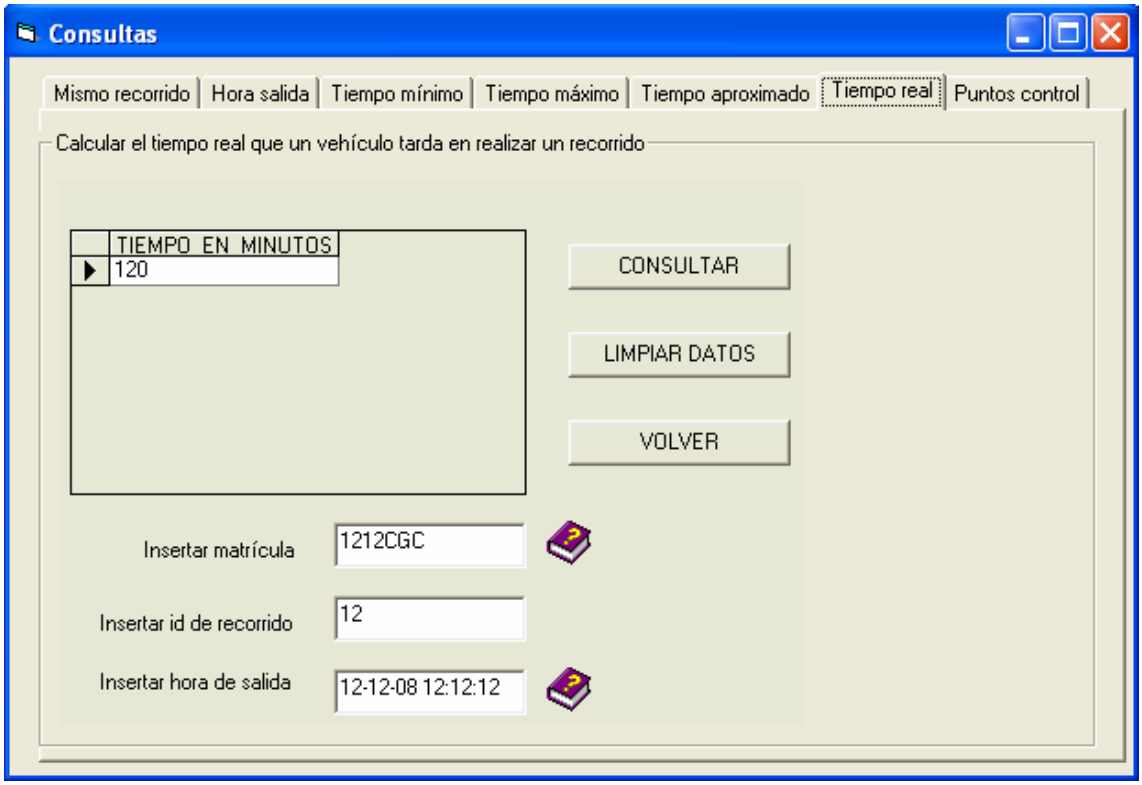

Ilustración 136 : Resultados consulta Tiempo real

En el caso en que no se encuentren datos que correspondan con los datos introducidos se mostrara la siguiente pantalla.

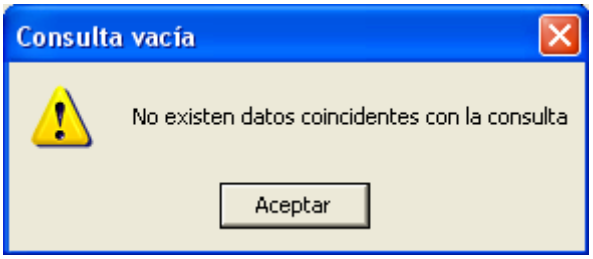

Ilustración 137 : Consulta vacía

6.4.7 *Puntos control*

En la consulta *Puntos control* se ofrece la posibilidad de consultar los puntos de control en los que un vehículo en concreto ha sobrepasado la velocidad permitida en dicho punto. Estos datos se corresponden con un recorrido en concreto, por lo que se deben insertar los datos de la matrícula del vehículo, el identificador del recorrido y la hora de salida.

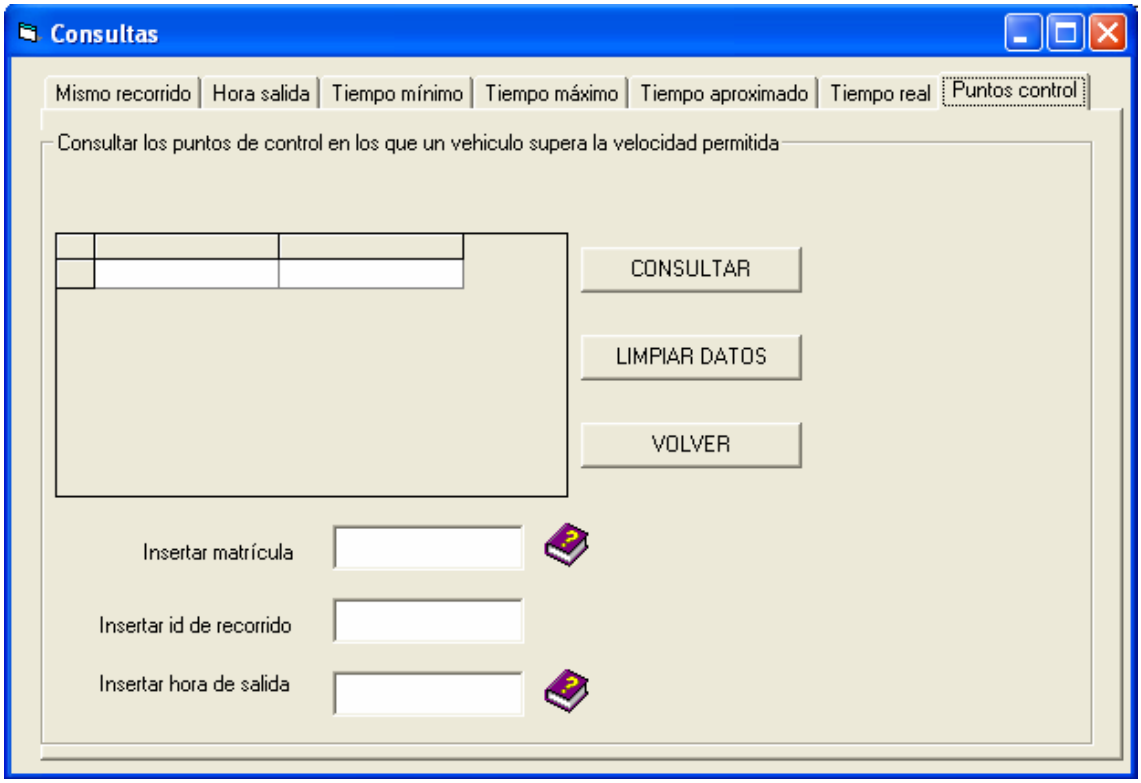

Ilustración 138 : Consulta Puntos Control

La siguiente pantalla muestra los datos obtenidos a partir de la consulta lanzada. Para este caso se ha introducido una matricula válida, un recorrido válido y una hora de salida válida.

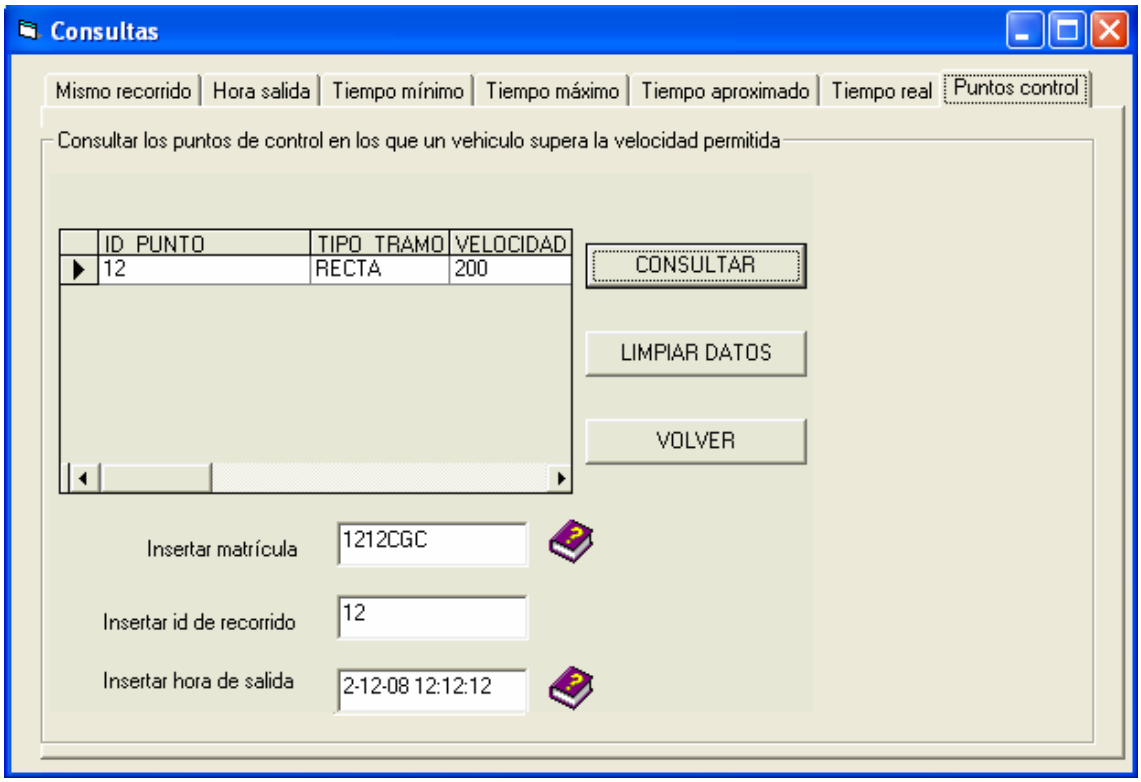

Ilustración 139 : Resultados consulta Puntos Control

En el caso en que no se encuentren datos que correspondan con los datos introducidos se mostrara la siguiente pantalla.

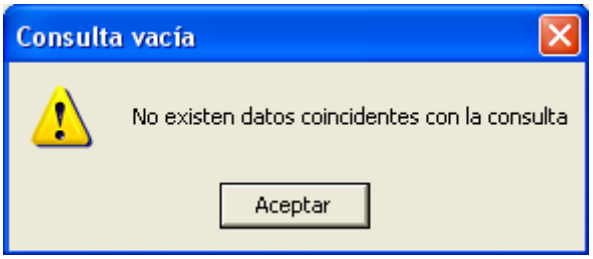

Ilustración 140 : Consulta vacía

7 PRESUPUESTO DEL PROYECTO

A continuación se ofrece el resumen de todas las horas invertidas en el presente proyecto:

Fase	Nº total de días	Nº total de horas
Aprendizaje de Oracle Spatial	10	80
Aprendizaje de Visual Basic 6.0	10	80
Instalación de Oracle 10g	1	8
Instalación de Visual Studio	½	4
Diseño de la base de datos (Modelos)	2	16
Creación de la base de datos	1	8
Diseño de la interfaz de la aplicación	2	16
Desarrollo de la interfaz de la aplicación	16	130
Pruebas	16	130
Redacción de la memoria	9	70
TOTAL	67,5	542

Tabla 16: Presupuesto del proyecto

Perfil en informática	Características	Salario Bruto Anual
Analista Programador	Experiencia entre 2 y 5 años	24000
Director de Proyecto	Experiencia de más de 10 años	40000

Tabla 17 : Salarios

Teniendo en cuenta que el número de horas por convenio son aproximadamente 1810 para el sector de Técnicos el coste hora quedaría:

Perfil en informática	Características	Coste Hora
Analista Programador	Experiencia entre 2 y 5 años	14
Director de Proyecto	Experiencia de más de 10 años	22

Tabla 18 : Coste por hora

Según la implicación que ha tenido cada uno y a partir de la información recogida en el punto anterior, es posible calcular el coste total en recursos humanos.

Miembro del equipo	Nº de horas	Coste total (€)
Analista Programador	542	7588
Director de Proyecto	36	792
	TOTAL (sin IVA)	8380
	TOTAL (con IVA)	9720,8

Tabla 19 : Coste total

8 CONCLUSIONES

En la introducción veíamos que el objetivo de este Proyecto de Fin de Carrera era demostrar el uso de las bases de datos espacio-temporales en un dominio. Creo que después de la finalización del proyecto ha quedado demostrado dicho funcionamiento.

Tengo que destacar que para mí lo más importante del proyecto, ha sido el tratamiento de los datos, tanto espaciales como temporales, debido principalmente a mi total desconocimiento del tema al principio del proyecto.

Por otro lado, el también desconocimiento del lenguaje de programación Visual Basic, ha sido otro de los aspectos más importantes del proyecto. La realización de la interfaz ha sido otro de los puntos que me han servido, junto con el entendimiento de las bases de datos espacio-temporales, como enriquecimiento, ya que ahora, después de realizar la aplicación he adquirido los conocimientos necesarios para poder enfrentarme a proyectos basados en estos temas.

Tengo que reconocer, que el tratamiento de las bases de datos con Oracle, ha resultado pesado, en el sentido de lentitud ya que la aplicación se realizó en un ordenador personal.

Para finalizar quisiera destacar que la realización de este proyecto me ha enriquecido enormemente, adquiriendo un conocimiento sobre las bases de datos espacio-temporales y entendiendo la problemática a la hora de implementar estos dominios de aplicación. Además ha sido una experiencia nueva y muy positiva realizar esta investigación tanto a nivel teórico como a nivel práctico.

9 LÍNEAS FUTURAS

Con el objeto de mejorar este proyecto en un futuro, se podrían realizar diversas modificaciones.

Probablemente de todas las modificaciones que se podrían realizar la más destacable sería una nueva interfaz, más extensa, como mayor número de posibilidades y con otro lenguaje que ofrezca un mayor número de opciones. Esta mejora, que aunque parezca que sólo se ocupa de la parte visual del proyecto, probablemente es una de las más importantes. Ya que en general, en muchos proyectos la interfaz es la parte a la que más importancia se le da.

En cuanto a funcionalidad (no a nivel de interfaz), la incorporación de sistemas GPS ayudaría a realizar una mayor precisión a la hora de controlar la situación exacta de los vehículos en cada momento.

Una mejora importante sería tener en cuenta las posibles condiciones adversas que se pueden producir de forma rutinaria en los trayectos de vehículos, ya sea un atasco, un accidente, o las condiciones meteorológicas. Esta mejora ayudaría mucho a tener una mayor precisión en los datos obtenidos a partir de las consultas.

Por último, otra mejora sería la utilización de índices en las tablas para mejorar así la velocidad de las consultas.

10 BIBLIOGRAFÍA Y ENLACES DE INTERÉS

Bases de datos espacio- temporales

[Kothuri, R., 2004]. Pro Oracle Spatial. Springe.

[Oracle, 2005]. Oracle Spatial User's Guide and Reference 10g Relase 2 (10.2). Oracle.

http://www.ing.ula.ve/~ibc/indObjET.pdf, documento web donde encontrar información sobre las bases de datos espacio-temporales.

Visual Basic 6.0

http://www.adrformacion.com/cursos/visual/leccion2/tutorial1.html, sitio web donde se puede encontrar información de visual basic.

http://www.recursosvisualbasic.com.ar/htm/tutoriales/control_list_box.htm, sitio web donde se puede encontrar información de visual basic.

http://otn.oracle.com, sitio web donde se podrán descargar los distintos productos Oracle, documentación, artículos de Oracle Corporation.

http://www.elguille.info, sitio donde encontrar información del lenguaje de programación VB 6 .0

11 ÍNDICE DE FIGURAS

12 ÍNDICE DE TABLAS